The Truth about Linear Regression

Cosma Rohilla Shalizi

Contents

Note to the Reader, on the Origin and Current State of the Text

This book is, currently, a very light revision of the notes I wrote for 36-401, "Modern Regression", at Carnegie Mellon University in the fall of 2015 (`http://www.stat.cmu.edu/~cshalizi/mreg/15`). I have tried to remove obvious errors, gross redundancies, and stuff that only mattered for that class, but no doubt such defects remain. More importantly, perhaps, to ensure continuity with previous iterations of the class, the text gives more emphasis to old-fashioned Gaussian-noise theory than I think it really warrants. (The don't-make-these-mistakes discussion of common misinterpretations of linear regression, especially in regard to causal inference, was ripped off from chapter 2 of Shalizi (forthcoming), which readers can find online.) The text also still has some references to other courses which either proceed or follow 36-401 in the CMU statistics curriculum: "225" is probability, "226" is mathematical statistics, and "402" is "advanced data analysis", comprising non-parametrics, multivariate analysis, and causal inference (Shalizi, forthcoming).

I hope to re-work these notes in the future, to present ideas like the bootstrap *first*, followed by the Gaussian-noise theory as a short-cut for a very special case, but that will have to wait for another day. In the meanwhile, I offer these notes to readers in the hope that they may be of some use. Feedback, and more especially corrections, will be gratefully received.

Finally, if you find these notes on a website *other* than my own, I would appreciate hearing about it; I have not given permission for such re-distribution, and the URL which should be printed on the first page of each chapter will always have the most up-to-date and corrected version of the full text, for free.

Introduction

"Regression" means predicting one continuous random variable from others, which might be continuous or might not. Most forms of regression work by "smoothing" observed values of the variable being predicted, meaning their predictions are weighted averages. Linear regression is the special case where we smooth the observed data on to a perfectly straight line (or plane, etc.), and then extrapolate the line to make predictions at new points. These predictions will be "unbiased", i.e., free from systematic errors, if and only if the relationship between the predictors and what we're predicting really is exactly linear.

Linear regression is widely used throughout the sciences and technology. The biggest reason for this is historical. Linear regression was invented about two hundred years ago, and up through, say, the 1970s any other kind of regression was generally too computationally expensive to be practical. Partly as a result, a huge body of theory has grown up around this rather weak prediction method. Most of this mathematical theory makes assumptions which are rarely even close to true in practice, starting with exact linearity and growing more implausible from there. Alongside the theory has grown a body of lore, or even mythology, which has more to do with what scientists *hope* they could get from data analysis than what it can actually accomplish.

Since about 1980, linear regression has become largely technologically obsolete, with three important exceptions. First, there are a handful of situations where there are genuine scientific reasons to use linear models. Second, there are applications which put a huge premium on computational simplicity. Third, sometimes we *want* biased, systematically-wrong predictions, because less-biased predictors will be more sensitive to noise when there isn't a lot of data. (That is, with small samples we may trade variance for bias.)

Beyond these rather special cases, there are two good reasons to study linear regression. One is that many more flexible and useful methods of regression can be seen, mathematically, as extensions or generalizations of linear regression. Even when that's not the case, many of the ideas behind better methods — such as smoothing, the bias-variance trade-off, cross-validation, regularization, etc. — can be illustrated for linear models using very basic math, giving intuition that still holds in more complex cases. (We can talk about linear functions rather than reproducing kernel Hilbert spaces.) The second and lesser reason is that, for the historical reasons I mentioned above, linear regression is a key part of the shared culture of data analysis, and understanding it — even the myths about it — is important if you want to participate in that culture.

These notes try to introduce the key parts of the theory and practice of linear regression from the perspective sketched above, i.e., as though the last forty years of statistics had actually happened. My starting point is the idea that linear regression is a method of prediction based on smoothing observed data on to a line. I thus emphasize methods and results about prediction which are agnostic about whether or not reality is truly linear. I give less emphasis to ideas which presume that some linear model is exactly true, and that what we care about is describing that true linear model very carefully. I also emphasize robust, computation-intensive techniques like cross-validation and the bootstrap, over exact formulas which make very fragile assumptions, though I do cover the latter for the sake of completeness. I also try to provide debunkings of common misunderstandings, exaggerations and outright myths.

Assumed Background

This book grew out of the notes for a class on linear regression taught to third and fourth year undergraduates at Carnegie Mellon University, majoring in statistics and related areas. The pre-requisites for the class were a first course in mathematical statistics , which in turn required a class in probability theory (which in turn required calculus), and a class in linear algebra. Most students had also taken a class in basic data analysis. Most had some exposure to the R programming language / statistical computing environment, and R was used extensively in the class. This book accordingly makes *no* attempt to explain the background ideas of mathematical statistics, probability theory, linear algebra, multi-variable calculus, or the basic use of R.

Part I
Regression with One Predictor

Chapter 1

Optimal Prediction (with Refreshers)

Regression analysis is about investigating *quantitative*, *predictive* relationships between variables. It's about situations where there is some sort of link, tie or relation between two (or more) variables, so if we know the value of one of them, it tells us something about the other. The concrete sign of this is that knowledge of one variable lets us *predict* the other — predict the target variable better than if we didn't know the other. Pretty much everything we are going to do in this class is about crafting predictive mathematical models, seeing whether such models really have any predictive power, and comparing their predictions. Before we get into the issues of statistics and data analysis, it will help us to think what *optimal* prediction would look like, if we somehow knew all the probability distributions of all our variables.

§1.1 refers to many concepts from probability (reviewed in §1.2) and statistical inference (reviewed in §1.3).

1.1 Statistical Prediction and the Optimal Linear Predictor

1.1.1 Predicting a Random Variable from Its Distribution

Suppose we want to guess the value of a random variable Y. Since we don't feel comfortable with the word "guess", we call it a "prediction" instead. What's the best guess we can make?

We need some way to measure how good a guess is. Say our guess is m. The difference $Y - m$ should somehow be small. If we don't care about positive more than negative errors, it's traditional to care about the squared error, $(Y - m)^2$. Since Y is random, this will fluctuate; let's look at its expected value,

$$\mathbb{E}\left[(Y-m)^2\right] \tag{1.1}$$

We will call this the **mean squared error** of m, $MSE(m)$.

From the definition of variance,

$$MSE(m) = \mathbb{E}\left[(Y-m)^2\right] = (\mathbb{E}[Y-m])^2 + \mathrm{Var}[Y-m] \tag{1.2}$$

The first term is the squared bias of estimating Y with m; the second term is the variance of $Y-m$. Mean squared error is bias (squared) plus variance. This is the simplest form of the **bias-variance decomposition**, which is one of the central parts of statistics.

Now remember that $\mathrm{Var}[Y-m] = \mathrm{Var}[Y]$, so

$$\begin{aligned} MSE(m) &= (\mathbb{E}[Y-m])^2 + \mathrm{Var}[Y] && (1.3)\\ &= (\mathbb{E}[Y]-m)^2 + \mathrm{Var}[Y] && (1.4) \end{aligned}$$

where the second line uses the linearity of expectations.

We would like to pick m to make this small, to minimize it (Figure 1.1). The variance term is irrelevant to making this small, since it's the same no matter what m is. (Remember, $\mathrm{Var}[Y]$ is about the true distribution of Y, but m is just our guess.) It should therefore play no role in the minimization.

Remember from basic calculus that one way to find the minimum[1] of a function is to take the derivative, set it to zero, and solve for the minimizing argument to the function. Here what we want to minimize is $MSE(m)$ and the argument is m, so

$$\frac{dMSE(m)}{dm} = \frac{d}{dm}\left[\mathrm{Var}[Y] + (\mathbb{E}[Y]-m)^2\right] \tag{1.5}$$

is the derivative we need to work out and set to zero. So, using the chain rule,

$$\frac{dMSE(m)}{dm} = \frac{d\mathrm{Var}[Y]}{dm} + 2(\mathbb{E}[Y]-m)\left(\frac{d\mathbb{E}[Y]}{dm} - \frac{dm}{dm}\right) \tag{1.6}$$

Changing the prediction we make, m, doesn't do anything to the true distribution of Y, so $d\mathrm{Var}[Y]/dm = d\mathbb{E}[Y]/dm = 0$, and we've got

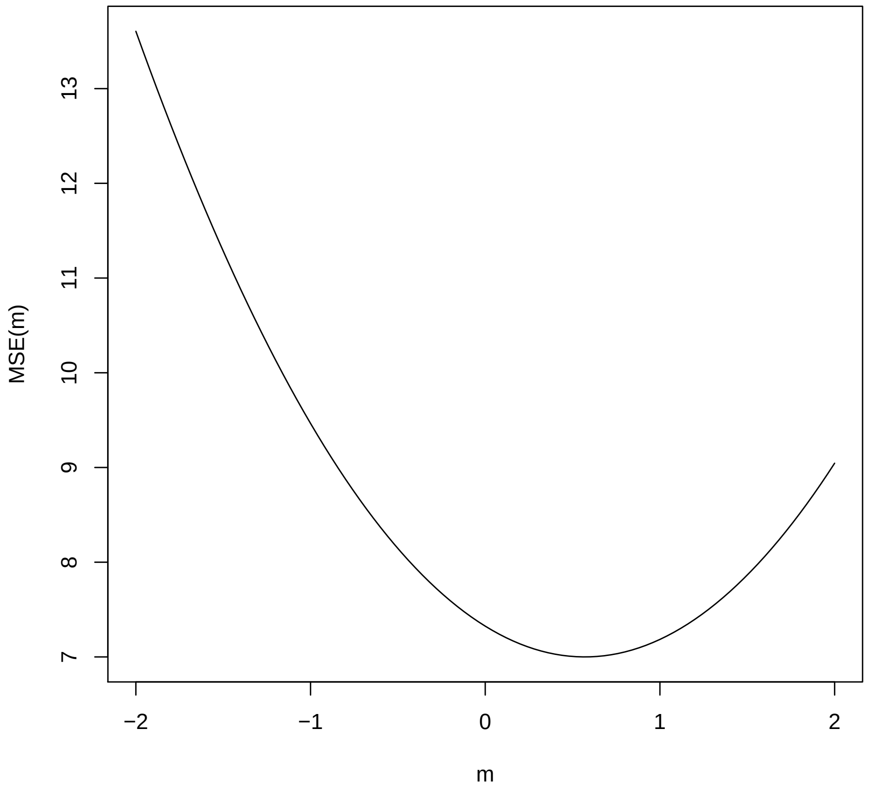
$$\frac{dMSE(m)}{dm} = -2(\mathbb{E}[Y]-m) \tag{1.7}$$

Say this is zero at $m = \mu$, and solve for μ:

$$\begin{aligned} -2(\mathbb{E}[Y]-\mu) &= 0 && (1.8)\\ \mathbb{E}[Y]-\mu &= 0 && (1.9)\\ \mathbb{E}[Y] &= \mu && (1.10) \end{aligned}$$

In other words, the best one-number guess we could make for Y is just its expected value.

[1] Or maximum; but here it's a minimum. (How could you check this, if you were worried that I was wrong?)



```
curve(7 + (0.57 - x)^2, from = -2, to = 2, xlab = "m", ylab = "MSE(m)")
```

FIGURE 1.1: *Mean squared error* $\mathbb{E}\left[(Y-m)^2\right]$ *as a function of the value* m *which we predict, when* $\mathbb{E}[Y]=0.57$, $\mathrm{Var}[Y]=7$. *(The text below the plot shows the R command used to make it.*

1.1.2 Predicting One Random Variable from Another

Now imagine we have two random variables, say X and Y. We know X and would like to use that knowledge to improve our guess about Y. Our guess is therefore a function of x, say $m(x)$. We would like $\mathbb{E}\left[(Y-m(X))^2\right]$ to be small.

We can use conditional expectations to reduce this problem to the one already solved.

$$\mathbb{E}\left[(Y-m(X))^2\right] = \mathbb{E}\left[\mathbb{E}\left[(Y-m(X))^2 \mid X\right]\right] \tag{1.11}$$

For each possible value x, the optimal value $\mu(x)$ is just the conditional mean, $\mathbb{E}[Y|X=x]$. The optimal function just gives the optimal value at each point:

$$\mu(x) = \mathbb{E}[Y \mid X = x] \tag{1.12}$$

This $\mu(x)$ is called the (true, optimal, or population) **regression function** (of Y on X). If we are interested in the relationship between Y and X, this is what we would really like to know, or one of the things we'd really like to know.

Unfortunately, in general $\mu(x)$ is a really complicated function, for which there exists no nice mathematical expression. The Ancestors, then, in their wisdom decided to ask "what is the best prediction we can make which is also a *simple* function of x?" In other words, they substituted a deliberately simplified *model* of the relationship for the actual relationship.

1.1.3 The Optimal Linear Predictor

Many people regard linear functions as especially simple[2], so let us now ask "What is the optimal prediction we can make which is *linear* in X?" That is, we restrict our prediction function $m(x)$ to have the form $b_0 + b_1 x$. (To be really pedantic, that's an "affine" rather than a "linear" function.)

The mean squared error of the linear model $b_0 + b_1 x$ is now a function of two arguments, b_0 and b_1. Let's re-write it to better separate the contributions from those arguments (which we control) and the contributions from the distribution of X and Y (which are outside our control).

$$\begin{aligned}
& MSE(b_0, b_1) \\
&= \mathbb{E}\left[(Y-(b_0+b_1X))^2\right] && (1.13)\\
&= \mathbb{E}\left[Y^2\right] - 2b_0\mathbb{E}[Y] - 2b_1\mathbb{E}[XY] + \mathbb{E}\left[(b_0+b_1X)^2\right] && (1.14)\\
&= \mathbb{E}\left[Y^2\right] - 2b_0\mathbb{E}[Y] - 2b_1(\mathrm{Cov}[X,Y] + \mathbb{E}[X]\mathbb{E}[Y]) && (1.15)\\
&\quad + b_0^2 + 2b_0b_1\mathbb{E}[X] + b_1^2\mathbb{E}\left[X^2\right] \\
&= \mathbb{E}\left[Y^2\right] - 2b_0\mathbb{E}[Y] - 2b_1\mathrm{Cov}[X,Y] - 2b_1\mathbb{E}[X]\mathbb{E}[Y] && (1.16)\\
&\quad + b_0^2 + 2b_0b_1\mathbb{E}[X] + b_1^2\mathrm{Var}[X] + b_1^2(\mathbb{E}[X])^2
\end{aligned}$$

[2]Actually being precise about "how complicated is this function?" is a surprisingly hard matter. (To appreciate this, think about how a straight line may seem like a simple function, but so does a step function, and yet you need a lot of little steps to approximate a straight line...) Resolving this leads to some very deep mathematics (Badii and Politi, 1997; Li and Vitányi, 1997).

(See §1.2 for the identities I'm using above.)

We minimize again by setting derivatives to zero; we now need to take two partial derivatives, which will give us two equations in two unknowns.

$$\frac{\partial \mathbb{E}\left[(Y-(b_0+b_1X))^2\right]}{\partial b_0} = -2\mathbb{E}[Y]+2b_0+2b_1\mathbb{E}[X] \quad (1.17)$$

$$\frac{\partial \mathbb{E}\left[(Y-(b_0+b_1X))^2\right]}{\partial b_1} = -2\mathrm{Cov}[X,Y]-2\mathbb{E}[X]\mathbb{E}[Y]+2b_0\mathbb{E}[X] + 2b_1\mathrm{Var}[X]+2b_1(\mathbb{E}[X])^2 \quad (1.18)$$

We'll call the optimal value of b_0 and b_1, the ones where these derivatives are exactly 0, β_0 and β_1.

The first equation is simpler, so we use it to find β_0 in terms of β_1:

$$\beta_0 = \mathbb{E}[Y] - \beta_1\mathbb{E}[X] \quad (1.19)$$

Some points about this equation:

- In words, it says that the optimal intercept (β_0) makes sure that the line goes through the mean Y value at the mean X value. (To see this, add $\beta_1\mathbb{E}[X]$ to both sides.)
- It's often helpful to sanity-check our math by making sure that the units balance on both sides of any equation we derive. Here, β_0 should have the same units as Y, and the right-hand side of this formula does, because β_1 has the units of Y/X.
- If the variables were "centered", with $\mathbb{E}[X] = \mathbb{E}[Y] = 0$, we'd get $\beta_0 = 0$.

Now we plug this in to the other equation:

$$0 = -\mathrm{Cov}[X,Y]-\mathbb{E}[X]\mathbb{E}[Y]+\beta_0\mathbb{E}[X]+\beta_1\mathrm{Var}[X] + \beta_1(\mathbb{E}[X])^2 \quad (1.20)$$

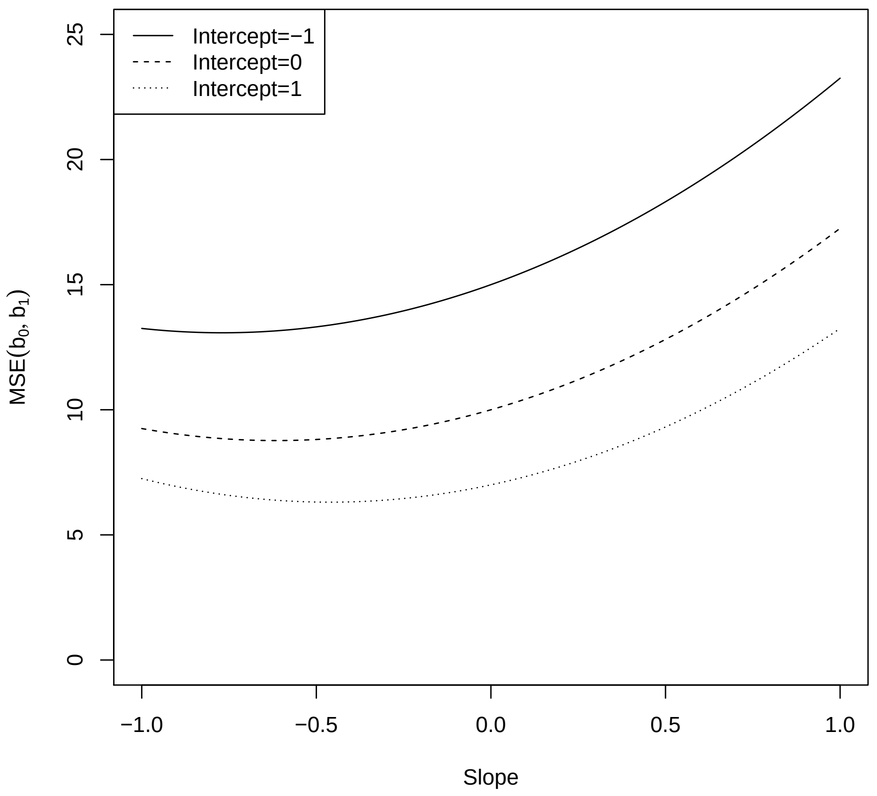
$$= -\mathrm{Cov}[X,Y]-\mathbb{E}[X]\mathbb{E}[Y] + (\mathbb{E}[Y]-\beta_1\mathbb{E}[X])\mathbb{E}[X]+\beta_1\mathrm{Var}[X]+\beta_1(\mathbb{E}[X])^2 \quad (1.21)$$

$$= -\mathrm{Cov}[X,Y]+\beta_1\mathrm{Var}[X] \quad (1.22)$$

$$\beta_1 = \mathrm{Cov}[X,Y]/\mathrm{Var}[X] \quad (1.23)$$

Some notes:

- In words, the optimal slope is the ratio between the covariance of X and Y, and the variance of X. The slope increases the more X and Y tend to fluctuate together, and gets pulled towards zero the more X fluctuates period.
- You can apply the sanity check of seeing whether this gives the right units for β_1. (Spoiler: it does.)



```
mse <- function(b0, b1, E.Y.sq = 10, E.Y = 2, Cov.XY = -1, E.X = -0.5, Var.X = 3) {
    E.Y.sq - 2 * b0 * E.Y - 2 * b1 * Cov.XY - 2 * b1 * E.X * E.Y + b0^2 + 2 *
        b0 * b1 * E.X + Var.X * b1^2 + (E.X * b1)^2
}
curve(mse(b0 = -1, b1 = x), from = -1, to = 1, lty = "solid", ylim = c(0, 25),
    xlab = "Slope", ylab = expression(MSE(b[0], b[1])))
curve(mse(b0 = 0, b1 = x), add = TRUE, lty = "dashed")
curve(mse(b0 = 1, b1 = x), add = TRUE, lty = "dotted")
legend("topleft", legend = c("Intercept=-1", "Intercept=0", "Intercept=1"),
    lty = c("solid", "dashed", "dotted"))
```

FIGURE 1.2: *Mean squared error of linear models with different slopes and intercepts, when* $\mathbb{E}[X] = -0.5$, $\mathrm{Var}[X] = 3$, $\mathbb{E}[Y] = 2$, $\mathbb{E}\left[Y^2\right] = 10$, $\mathrm{Cov}[X,Y] = -1$. *Each curve represents a different intercept* b_0 *in the linear model* $b_0 + b_1 x$ *for* Y.

- The expected values $\mathbb{E}[X]$ and $\mathbb{E}[Y]$ play no role in the formula for β_1 — only the variance and covariance matter, and they don't change when we add or subtract constants. In particular, the optimal slope doesn't change if we use instead $Y - \mathbb{E}[Y]$ and $X - \mathbb{E}[X]$.

The line $\beta_0 + \beta_1 x$ is the **optimal regression line** (of Y on X), or the **optimal linear predictor** (of Y from X).

Important Morals

1. At no time did we have to assume that the relationship between X and Y really is linear. We have derived the optimal linear approximation to the true relationship, whatever that might be.
2. The best linear approximation to the truth can be awful. (Imagine $\mathbb{E}[Y|X=x] = e^x$, or even $= \sin x$.) There is no general reason to think linear approximations *ought* to be good.
3. At not time did we have to assume anything about the marginal distributions of the variables[3], or about the joint distribution of the two variables together[4]
4. At no time did we have to assume anything about the fluctuations Y might show around the optimal regression line — that the fluctuations are Gaussian, or symmetric, or that they just add on to the regression line, etc.
5. In general, changing the distribution of X will change the optimal regression line, even if $\mathbb{P}(Y|X=x)$ doesn't change. This is because changing the distribution of X will (generally) change both $\mathrm{Cov}[X,Y]$ and $\mathrm{Var}[X]$, and the changes won't (generally) cancel out.
6. At no time did we have to assume that X came before Y in time, or that X causes Y, or that X is known precisely but Y only noisily, etc. It may be more *interesting* to model Y as a linear function of X under those circumstances, but the math doesn't care about it at all.

I will expand on that first two points a little. There is a *sort* of reason to think that linear models should work generally, which contains a kernel of truth, but needs to be used carefully.

The true regression function, as I said, is $\mu(x)$. Suppose that this is a smooth function, so smooth that we can expand it in a Taylor series. Pick then any particular value x_0. Then

$$\mu(x) = \mu(x_0) + (x - x_0)\left.\frac{d\mu}{dx}\right|_{x=x_0} + \frac{1}{2}(x-x_0)^2 \left.\frac{d^2\mu}{dx^2}\right|_{x=x_0} + \ldots \tag{1.24}$$

[3]OK, to be pedantic, we had to assume that $\mathbb{E}[X]$, $\mathbb{E}[Y]$, $\mathrm{Var}[X]$ and $\mathrm{Var}[Y]$ were all well-defined and $\mathrm{Var}[X] > 0$.

[4]Except, to keep being pedantic, that $\mathrm{Cov}[X,Y]$ was well-defined.

Because it's tiresome to keep writing out the derivatives in this form, I'll abbreviate them as μ', μ'', etc.

For x close enough to x_0, we can get away with truncating the series at first order,

$$\mu(x) \approx \mu(x_0) + (x - x_0)\mu' \tag{1.25}$$

and so we could identify that first derivative with the optimal slope β_1. (The optimal intercept β_0 would depend on $\mu(x_0)$ and the distribution of $x - x_0$.) How close is enough? Close enough that all the other terms don't matter, so, e.g., the quadratic term has to be negligible, meaning

$$|x - x_0|\mu' \gg |x - x_0|^2\mu''/2 \tag{1.26}$$

$$2\mu'/\mu'' \gg |x - x_0| \tag{1.27}$$

Unless the function is really straight, therefore, any linear approximation is only going to be good over very short ranges.

It *is* possible to do a lot of "local" linear approximations, and estimate $\mu(x)$ successfully that way — in fact, we'll see how to do that in 402 (or read Simonoff 1996). But a justification for a *global* linear model, this is weak.

A better justification for using linear models is simply that they are *computationally* convenient, and there are many situations where computation is at a premium. If you have huge amounts of data, or you need predictions very quickly, or your computing hardware is very weak, getting a simple answer can be better than getting the *right* answer. In particular, this is a rationale for using linear models to make *predictions*, rather than than for caring about their *parameters*.

All of that said, we are going to spend most of this course talking about doing inference on the parameters of linear models. There are a few reasons this is not *totally* perverse.

- The theory of linear models is a special case of the more general theory which covers more flexible and realistic models. But precisely because it is such a special case, it allows for many simplifying short-cuts, which can make it easier to learn, especially without advanced math. (We can talk about points and lines, and not about reproducing-kernel Hilbert spaces.) Learning linear models first is like learning to swim in a shallow pool, rather than in the ocean with a gorgeous reef, deceptive currents, and the occasional shark. (By the end of the year, you will know how to dive with small sharks.)

- Because linear models are so simple, for most of the last two hundred odd years they were the only sort of statistical model people could actually *use*. This means that lots of applications of statistics, in science, in policy and in industry, has been done on linear models. It also means that lots of consumers of *statisticians*, in science, in policy and in industry, expect linear models. It is therefore important that you understand thoroughly both how they work and what their limitations are.

Throughout the rest of the course, we are going to tack back and forth between treating the linear model as exactly correct, and treating it as just a more-or-less convenient, more-or-less accurate approximation. When we make the stronger assumption

FIGURE 1.3: *Statistician (right) receiving population moments from the Oracle (left).*

that the linear model is right, we will be able to draw stronger conclusions; but these will not be much more secure than that assumption was to start with.

1.1.4 Probability versus Statistics

Everything I've gone over so far is purely mathematical. We have been pretending (Figure 21.5) that we have gone to the Oracle, and in a mystic trance they have revealed to us the full joint probability distribution of X and Y, or at least the exact values of all of their moments. As in many mathematical problems, therefore, we have idealized away everything inconvenient. In reality, we never know the full probability distribution of the variables we are dealing with[5]. Rather than exact knowledge from the Oracle, we have only a limited number of noisy samples from the distribution. The *statistical* problem is that of drawing inferences about the ideal predictor from this unpromising material.

1.2 Reminders from Basic Probability

The expectation (or expected value) of a continuous random variable X with probability density function $p(x)$ is

$$\mathbb{E}[X] = \int x p(x) dx \tag{1.28}$$

while the expectation of a discrete random variable with probability mass function $p(x)$ is

$$\mathbb{E}[X] = \sum_x x p(x) \tag{1.29}$$

[5]Even the idea that the variables we see *are* randomly generated from a probability distribution is a usually-untestable assumption.

(Because everything is parallel for the discrete and continuous cases, I will not keep writing out both forms; having tossed a coin to decide which, I will just write out the integrals.)

The expectation of any function of a random variable $f(X)$ is

$$\mathbb{E}[f(X)] = \int f(x)p(x)dx \tag{1.30}$$

(Of course, $f(X)$ has its own distribution, with a density we might call p_f; can you prove that that $\int f(x)p(x)dx = \int h p_f(h)dz$?)

$X - \mathbb{E}[X]$ is the **deviation** or **fluctuation** of X from its expected value.

The **variance** of X is

$$\text{Var}[X] = \mathbb{E}\left[(X - \mathbb{E}[X])^2\right] \tag{1.31}$$

The **covariance** of X and Y is

$$\text{Cov}[X,Y] = \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])] \tag{1.32}$$

The covariance is positive when X and Y tend to be above or below their expected values together, and negative if one of them having a positive fluctuation tends to go with the other having a negative fluctuation.

1.2.1 Algebra with Expectations, Variances and Covariances

We're going to deal a lot with expectation values, variances and covariances. There are some useful bits of algebra about these, which I will now remind you of. You will commit them to memory (either deliberately or because you'll use them so often).

1. *Linearity of expectations*

$$\mathbb{E}[aX + bY] = a\mathbb{E}[X] + b\mathbb{E}[Y] \tag{1.33}$$

2. *Variance identity*

$$\text{Var}[X] = \mathbb{E}\left[X^2\right] - (\mathbb{E}[X])^2 = \mathbb{E}\left[(X - \mathbb{E}[X])^2\right] \tag{1.34}$$

3. *Covariance identity*

$$\text{Cov}[X,Y] = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y] = \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])] \tag{1.35}$$

4. *Covariance is symmetric*

$$\text{Cov}[X,Y] = \text{Cov}[Y,X] \tag{1.36}$$

5. *Variance is covariance with itself*

$$\text{Cov}[X,X] = \text{Var}[X] \tag{1.37}$$

6. *Variance is not linear*
$$\mathrm{Var}[aX+b] = a^2\mathrm{Var}[X] \tag{1.38}$$

7. *Covariance is not linear*
$$\mathrm{Cov}[aX+b, Y] = a\mathrm{Cov}[X,Y] \tag{1.39}$$

8. *Variance of a sum*
$$\mathrm{Var}[X+Y] = \mathrm{Var}[X] + \mathrm{Var}[Y] + 2\mathrm{Cov}[X,Y] \tag{1.40}$$

9. *Variance of a big sum*
$$\mathrm{Var}\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n}\sum_{j=1}^{n}\mathrm{Cov}\left[X_i, X_j\right] = \sum_{i=1}^{n}\mathrm{Var}[X_i] + 2\sum_{i=1}^{n-1}\sum_{j>i}\mathrm{Cov}\left[X_i, X_j\right] \tag{1.41}$$

10. *Law of total expectation*
$$\mathbb{E}[X] = \mathbb{E}[\mathbb{E}[X|Y]] \tag{1.42}$$
Remember: $\mathbb{E}[X|Y]$ is a function of Y; it's random.

11. *Law of total variance*
$$\mathrm{Var}[X] = \mathrm{Var}[\mathbb{E}[X|Y]] + \mathbb{E}[\mathrm{Var}[X|Y]] \tag{1.43}$$

12. *Independence implies zero covariance* If X and Y are independent, $\mathrm{Cov}[X,Y] = 0$. The reverse is *not* true; $\mathrm{Cov}[X,Y] = 0$ is even compatible with Y being a function of X.

1.2.2 Convergence

The Law of Large Numbers Suppose that $X_1, X_2, \ldots X_n$ all have the same expected value $\mathbb{E}[X]$, the same variance $\mathrm{Var}[X]$, zero covariance with each other. Then
$$\frac{1}{n}\sum_{i=1}^{n} X_i \rightarrow \mathbb{E}[X] \tag{1.44}$$

In particular, if the X_i all have the same distribution and are independent ("independent and identically distributed", IID) then this holds.

Note: There are forms of the law of large numbers which don't even require a finite variance, but they are harder to state. There are also ones which do not require constant means, ore even a lack of covariance among the X_i, but they are also harder to state.

Central limit theorem If the X_i are IID, then as $n \to \infty$, the distribution of $\frac{1}{n}\sum_{i=1}^{n} X_i$ approaches $\mathcal{N}(\mathbb{E}[X], \mathrm{Var}[X]/n)$, regardless of the distribution of the X_i.

Mathematically, it is nicer to have the limit that we're converging to not change with n, so this is often stated as

$$\sqrt{n}\frac{\overline{X}_n - \mathbb{E}[X]}{\mathrm{Var}[X]} \rightsquigarrow \mathcal{N}(0,1) \tag{1.45}$$

Note: There are versions of the central limit theorem which do *not* assume independent or identically distributed variables being averaged, but they are considerably more complicated to state.

1.3 Reminders from Basic Statistics: Estimation

We observe values $X_1, X_2, \ldots X_n$ from some distribution. We don't know the distribution, so we imagine writing it down with one or more unknown parameters, $f(x;\theta)$. A **statistic** is a function of the data, and the data alone. An **estimator** is a statistic which takes a guess at the parameter θ, or some function of it, $h(\theta)$. (For instance we might want to estimate $\mathbb{E}\left[X^2\right] = \mu^2 + \sigma^2$.) We will generically write such an estimator as $\hat{\theta}_n$, with the hat to distinguish it from the true value of the parameter, and the subscript n to emphasize that it will change as we get more data.

An estimator is a random variable; it inherits its distribution from that of the data X_i. This is often called the **sampling distribution** of the estimator.

An estimator is **consistent** if $\hat{\theta}_n \to \theta$, whatever the true θ might be. An estimator which is not consistent is **inconsistent**, and usually not very good.

The **bias** of an estimator is $\mathbb{E}\left[\hat{\theta}_n - \theta\right] = \mathbb{E}\left[\hat{\theta}_n\right] - \theta$. An estimator is **unbiased** if its bias is zero for all θ.

An estimator also has a variance, $\mathrm{Var}\left[\hat{\theta}\right]$. The **standard error** of an estimator is its standard deviation, the square root of the variance. We give it the name "standard error" to remind ourselves that this is telling us about how precise our estimate is. N.B., there are more standard errors than just the standard error in the mean (see below).

An estimator cannot be consistent unless its standard error goes to zero as n grows. If both the standard error and the bias go to zero, that guarantees consistency, but there are exceptional circumstances where asymptotically biased estimators are still consistent.

Example: Sample Mean The expectation value $\mathbb{E}[X]$ is either a parameter of a distribution, or a function of the parameters. The sample mean $\overline{X}_n = n^{-1}\sum_{i=1}^{n} X_i$ is a statistic, since it is a function of the data alone. The sample mean can be used as an estimator of $\mathbb{E}[X]$, and is a natural choice for this role. If the X_i are IID, then the law of large numbers tells us that $\overline{X}_n$ is a consistent estimator of $\mathbb{E}[X]$. The central limit theorem tells us that the sampling distribution is asymptotically Gaussian.

It is easy to prove (so do so!) that $\mathbb{E}\left[\overline{X}_n\right] = \mathbb{E}[X]$, hence the mean is an unbiased estimator of the expected value. Notice that $\mathrm{Var}\left[\overline{X}_n\right] = \mathrm{Var}[X_1]/n$, which, as promised above, goes to zero as $n \rightarrow \infty$. The corresponding standard deviation is $\sigma/\sqrt{n}$, which is the "standard error in the mean". (Again, every estimator of every quantity has its own standard error, which is not just this.)

Example: Shrunken Sample Mean As an alternative estimator, consider $\frac{n}{n+\lambda}\overline{X}_n$, where you get to set the number $\lambda > 0$ (but then you have to use the same λ for all n). You should be able to convince yourself that (i) at every n and every λ, it has a strictly smaller variance than $\overline{X}_n$, and hence a strictly smaller standard error; (ii) it is a biased estimator of $\mathbb{E}[X]$, with a bias which depends on $\mathbb{E}[X]$, λ and n; (iii) for every $\mathbb{E}[X]$ and λ, the bias goes to zero as $n \rightarrow \infty$; (iv) it is a consistent estimator of $\mathbb{E}[X]$. This is an example of what is called a "shrinkage" estimator, where the obvious estimate is "shrunk" towards zero, so as to reduce variance.

Chapter 2

Introducing Statistical Modeling

2.1 Motivating

Let's start this off with a motivating example[1]. We'll begin by loading some data which comes from the Bureau of Economic Analysis, on the economic output of cities in the U.S. (`http://www.bea.gov/regional/gdpmetro/`).

```
bea <- read.csv("http://www.stat.cmu.edu/~cshalizi/mreg/15/lectures/03/bea-2006.csv")
```

For each city — more precisely, each "Metropolitan Statistical Area", which ignores legal divisions of cities and counties and instead is based on patterns of commuting — this records the name of the city, its population, its per-capita "gross metropolitan product" in 2006 (the total value of goods and services produced), and the share of the economy coming from four selected industries.

```
dim(bea)  # Should have 7 columns and 366 rows, and we do
## [1] 366   7
head(bea)  # Look at the beginning of the data
##                          MSA pcgmp    pop finance prof.tech     ict
## 1               Abilene, TX 24490 158700 0.09750        NA 0.01621
## 2                 Akron, OH 32890 699300 0.12940   0.05440      NA
## 3                Albany, GA 24270 163000 0.08217        NA 0.00708
## 4 Albany-Schenectady-Troy, NY 36840 850300 0.15780   0.09399 0.04511
## 5           Albuquerque, NM 37660 816000 0.15990   0.09978 0.20500
## 6            Alexandria, LA 25490 152200 0.09152   0.03790 0.01134
##   management
## 1         NA
## 2   0.054310
```

[1]I learned about this data from a paper by Bettencourt *et al.* (2007), but I think their data analysis is deeply flawed.

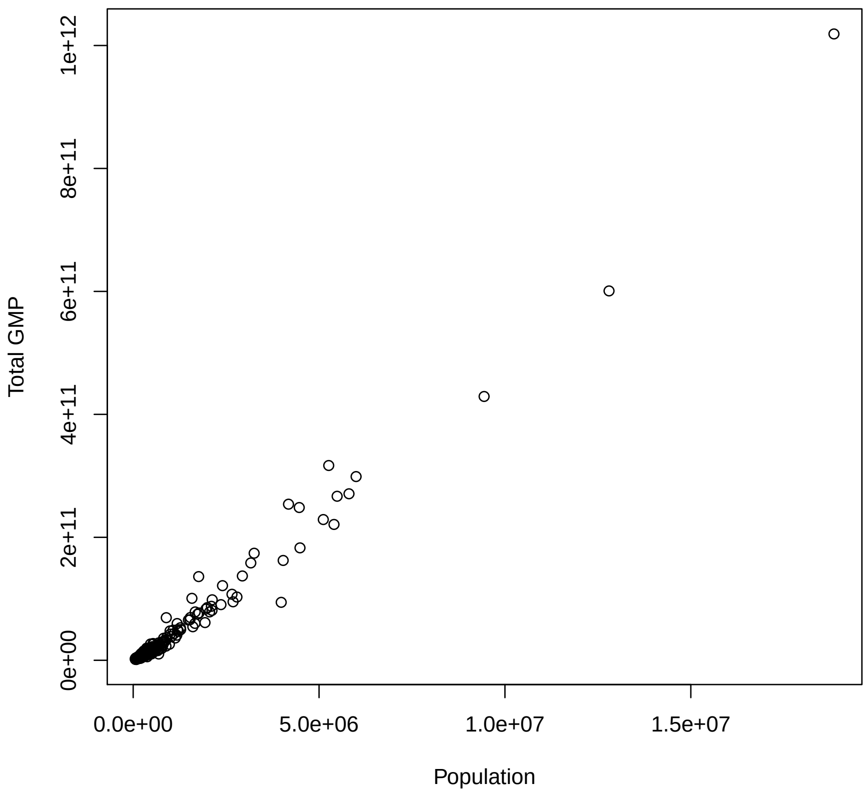
```
## 3         NA
## 4         NA
## 5   0.006509
## 6   0.015210
```

Let's add a new column, which records the *total* GMP, by multiplying the output per person by the number of people:

```
bea$gmp <- bea$pcgmp * bea$pop
```

And now let's look at this visually:

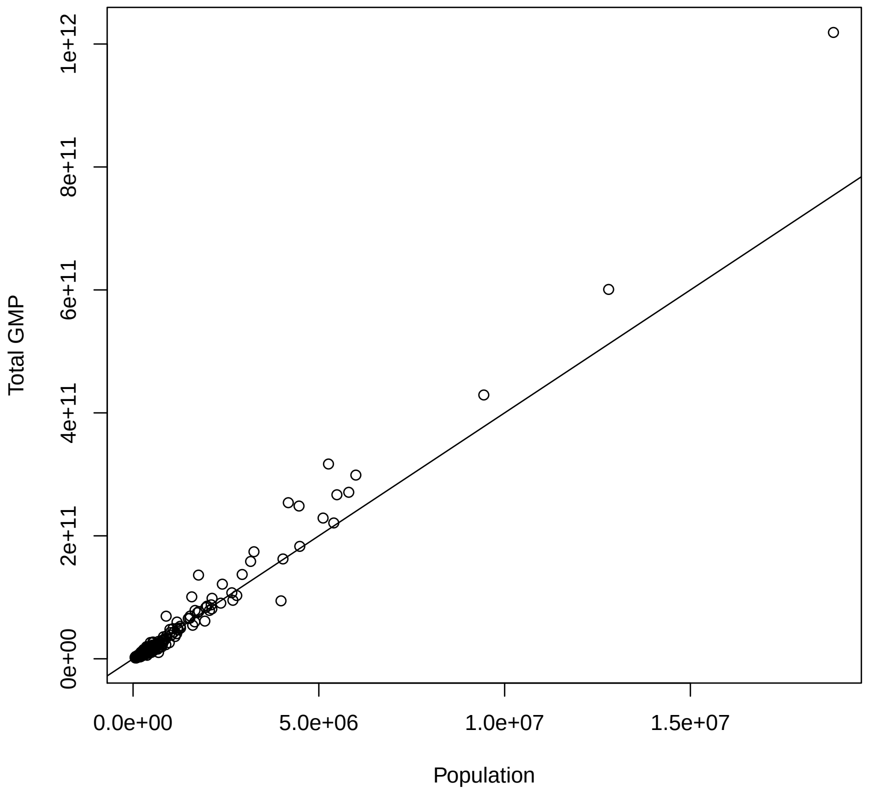
```
plot(gmp ~ pop, data = bea, xlab = "Population", ylab = "Total GMP")
```



The `plot` command, naturally enough, makes plots. The first argument to it tells it what to plot: here, we're telling it to plot `gmp` as a function of `pop`. (The tilde sign, is used in such "formulas" to indicate that what goes on the left is being treated as a function of what's on the right.) The next argument tells R where to look up the variables in the formula: in the data frame `bea` that we just loaded. The other two arguments give the axis some sensible labels.

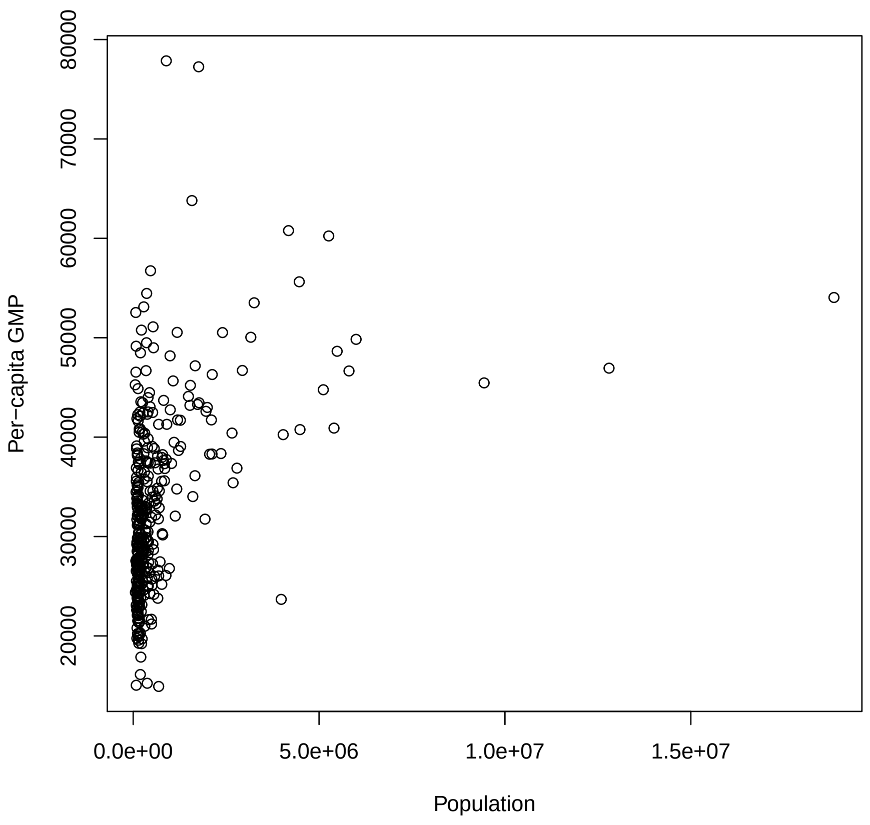
This plot shows, unsurprisingly, that larger cities have larger total economic outputs. What is more remarkable is how closely the points fall around a straight line. To see this, we'll re-plot the points, and use the function `abline` to add a straight line. To do this we need an intercept (the `a`) and a slope (the `b`). A reasonable guess for the intercept is `0` (since presumably a city with no inhabitants has no economy). One could reasonably guess the slope at 4×10^4 dollars/person, say by noticing the city with a population of about ten million (as it happens, Chicago) and seeing where it falls on the vertical axis.

```
plot(gmp ~ pop, data = bea, xlab = "Population", ylab = "Total GMP")
abline(a = 0, b = 40000)
```



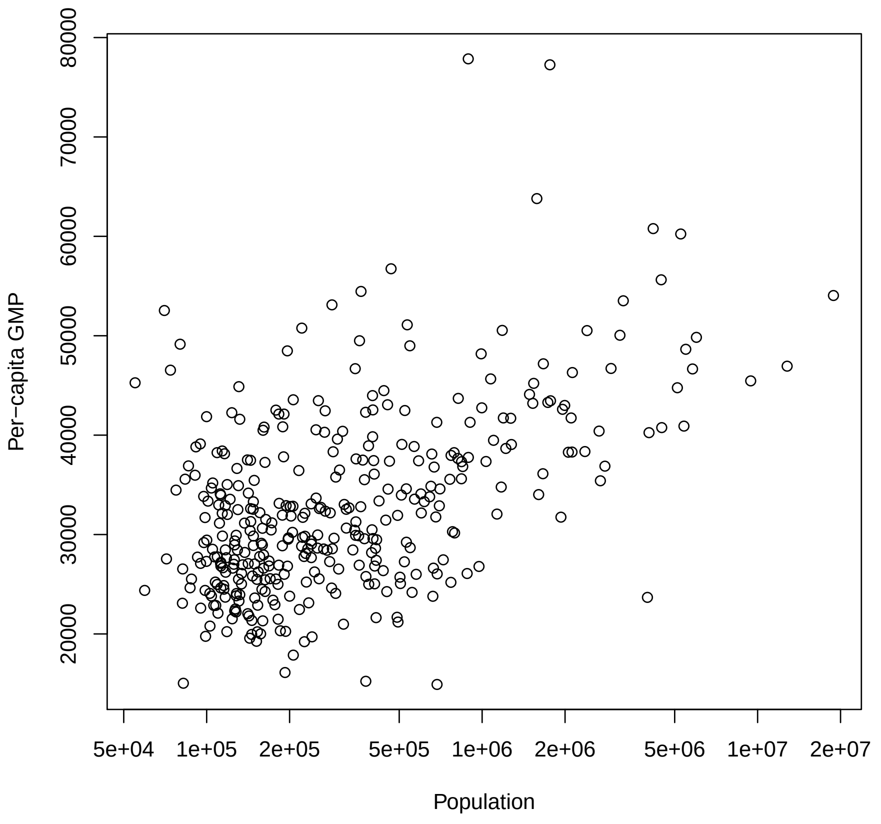
This isn't bad at all, but it looks like it's systematically too low for the larger cities. This is *suggestive* that there may be differences between the economies of large and small cities. Let's explore this by looking at the per-capita figures.

```
plot(pcgmp ~ pop, data = bea, xlab = "Population", ylab = "Per-capita GMP")
```



At this point, it becomes annoying that the larger cities in the US are so much larger than the small ones. By using a linear scale for the horizontal axis, we devote most of the plot to empty space around New York, Los Angeles and Chicago, which makes it harder to see if there is any trend. A useful trick is to switch to a logarithmic scale for that axis, where equal distances correspond to equal *multiples* of population.

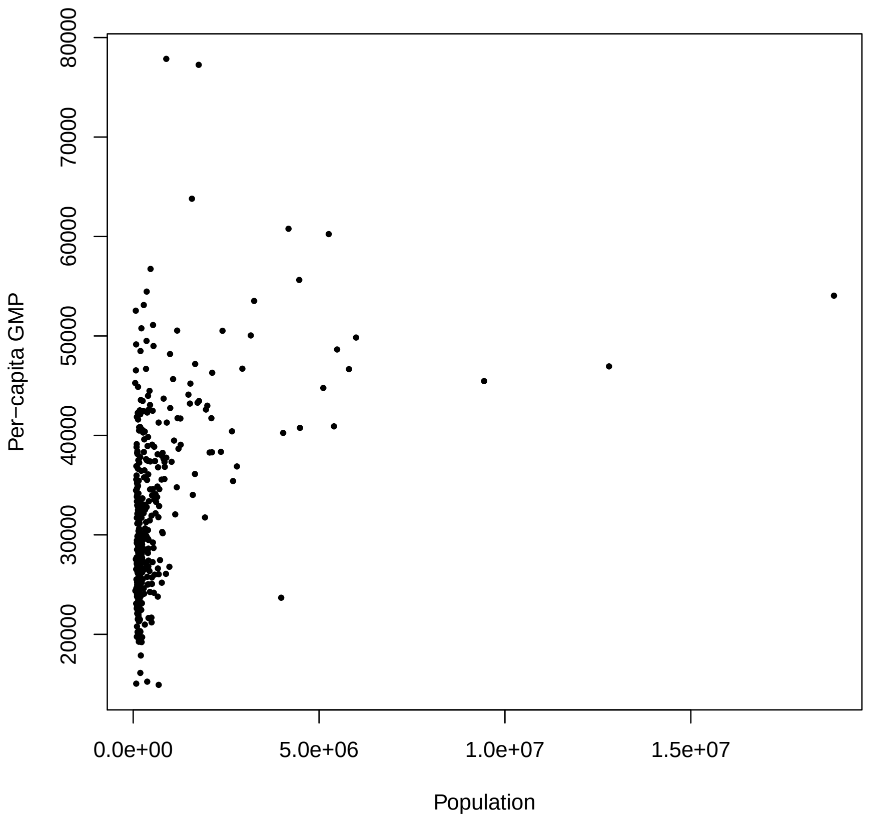
```
plot(pcgmp ~ pop, data = bea, xlab = "Population", ylab = "Per-capita GMP",
    log = "x")
```



Two things are noticeable from this plot. First, there is a wide range of per-capita GMP for smaller cities, which narrows as population grows. Second, there seems to be an increasing trend, or at least an increasing lower limit.

Let's restore the previous plot, but make it a bit less visually cluttered.

```
# alter the plotting symbol from a hollow circle to a filled dot (pch)
# shrink the plotting symbols by a factor of 0.5 (cex)
plot(pcgmp ~ pop, data = bea, xlab = "Population", ylab = "Per-capita GMP",
    pch = 19, cex = 0.5)
```



Let's now calculate our first regression line. R has a function for estimating linear models, with which we'll become very familiar:

```
lm(pcgmp ~ pop, data = bea)
##
## Call:
## lm(formula = pcgmp ~ pop, data = bea)
##
## Coefficients:
## (Intercept)          pop
##   3.128e+04    2.416e-03
```

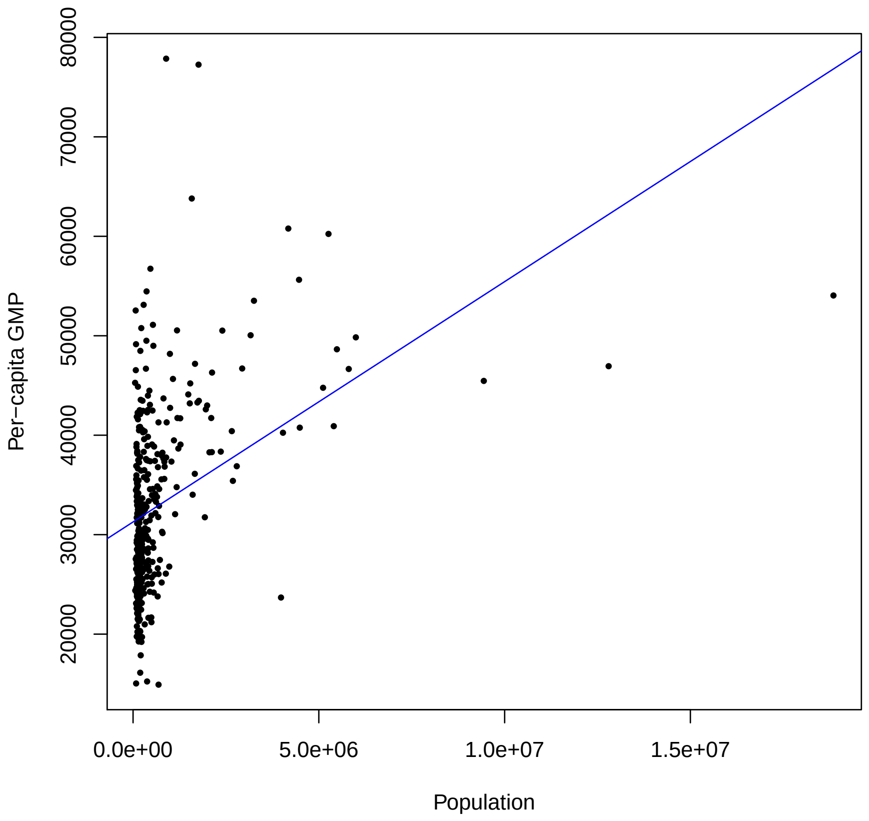
The first argument to `lm` is a formula, telling R which variable we're trying to predict (the one on the left, here `pcgmp`), and which variable we're trying to predict it from (the one on the right, here `pop`), and what data set to take those variables from (`bea` again)[2] R then *estimates* the coefficients of the best linear predictor — we will see

[2]Much more complicated formulas are possible, but this will do for now.

later how it does this — and returns those coefficients, along with a lot of other stuff which is invisible in this view.

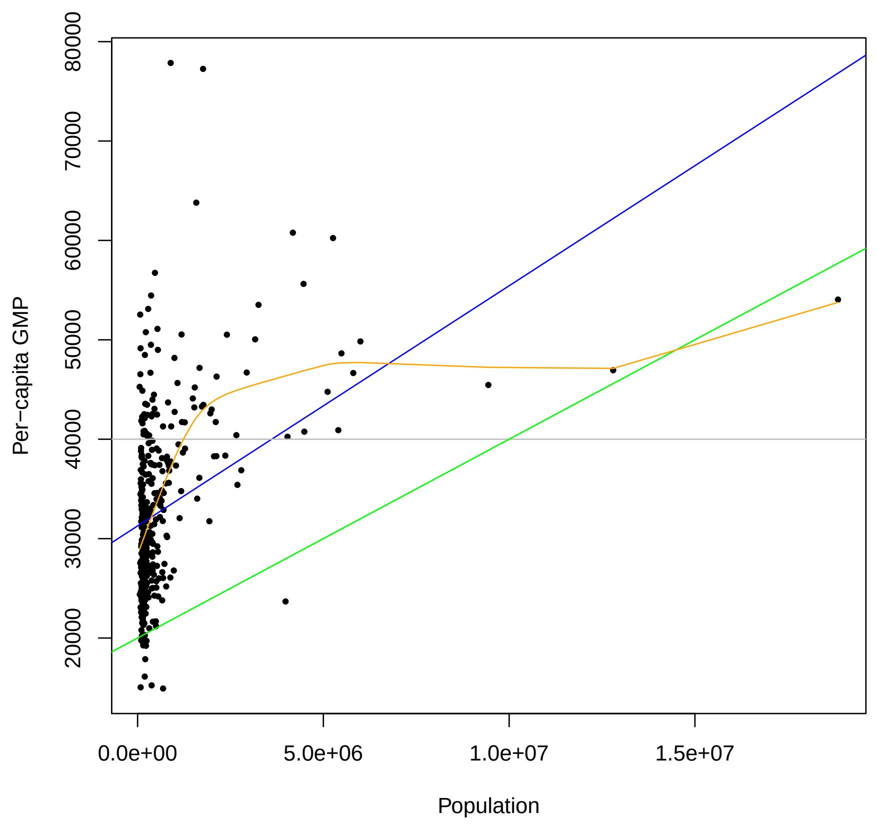
The `abline` command is smart enough to get an intercept and a slope from the output of `lm`, so we can use it to decorate the plot:

```
plot(pcgmp ~ pop, data = bea, xlab = "Population", ylab = "Per-capita GMP",
    pch = 19, cex = 0.5)
abline(lm(pcgmp ~ pop, data = bea), col = "blue")
```



But why should we believe that line? R does it, but I hope by this point in your life you don't think "the computer says it, so it must be right" is ever a good idea. Why prefer that blue line over this grey line, or the green one, or for that matter over the orange curve?

```
plot(pcgmp ~ pop, data = bea, xlab = "Population", ylab = "Per-capita GMP",
    pch = 19, cex = 0.5)
abline(lm(pcgmp ~ pop, data = bea), col = "blue")
abline(a = 40000, b = 0, col = "grey")
abline(a = 20000, b = 40000/2e+07, col = "green")
lines(smooth.spline(x = bea$pop, y = bea$pcgmp, cv = TRUE), col = "orange")
```



Why do we want to fit any lines here at all?

2.2 What Are Statistical Models For?

One basic use for these lines is as **summaries**. There is a lot of detail in those figures — each one requires $366 \times 2 = 732$ numbers. This is a lot of information to keep track of; it makes our heads hurt. In many situations, we'd rather ignore all of that precise detail and get away with a summary; we might want to *compress* the 732 numbers into just an intercept and a slope that describe the general trend or over-all shape of the data. There would be lots of ways to do that, and which ones would make sense would depend on how we want to use the compressed summary. We might, for instance, aim at being able to recover the original data with minimal error from the summary, so we'd want a line which came close to the original points. But we might also decide that it was more important for the line to come closer to large cities than small ones (since they contain so many more people), etc., etc.

There is nothing wrong with data compression — it is, in fact, one of the fundamental technologies of modern life — but I have little more to say about it (here). The one thing I will say is that *anything* you can calculate from the data could, in principle, be used as a summary. Even if the calculation was originally inspired by doing some sort of statistical inference, every statistic can be a descriptive statistic.

If we want to go beyond describing, summarizing or compressing the data, we enter the realm of **inference** — we try to reach out and extend our knowledge from the data we have, to other variables we have not measured, or not measured so directly. This is inherently somewhat risky, imprecise, and uncertain. In statistics, we aim not only to draw such inferences, but to say something about the level of risk, imprecision, and uncertainty which accompanies them.

You have, by this point in your educations, been thoroughly schooled in one way in which inferences can be subject to uncertainty: when the data is just a sample of a larger population, and we want to extrapolate from the sample to the population. In fact, many people get taught that this is the only sort of uncertainty statistics can handle.

If that were true, there would be no role for statistics in dealing with this data set. It isn't any sort of sample at all — every city in the US in 2006 really is in there. So why, then, does it seem wrong to say that the slope of the optimal linear predictor is *exactly* 0.0024162 dollars per year per person? There are at least two reasons.

One reason is that while we have measurements on the complete population, those measurements are themselves subject to error. In this case, while the people at the BEA try very hard to provide reliable numbers, their figures are the result of a complicated process, at which error can creep in at many points, from mistakes, accidents, deliberate lies, possibly-incorrect assumptions made at various stages, the use of random sampling in some steps, etc., etc.[3] Generally, just about every process of measurement is subject to some error: there are flaws in the measurement instrument or process, or we measure some imperfect proxy for what we're really interested in, or the measurement is perturbed by unrepeatable, irrelevant disturbances. In fact, much of the theory of mathematical statistics generally, and linear models specifically, was developed in the 19th century by astronomers and other physical scientists to quantify, and where possible reduce, the impacts of measurement error.

Some sources of measurement error are **systematic**: they tend to distort the measurement in predictable, repeatable ways. They may make the measured value larger than it should be, or smaller, or might shrink extreme values towards more mediocre ones, etc. Ideally, these systematic errors should be identified and modeled, so we can adjust for them. Other sources of error are (purely or merely) **statistical**: they are directionless and do not repeat, but the *distribution* of errors is predictable and stable; we can handle them with probability theory.

A second reason why it's reasonable to do statistical inference on a complete data

[3]This is, among other things, a reason to want to compress the data, rather than just memorizing it: some part of the data is just wrong.

set is that even the real values re somewhat accidental, and we'd like to know the general, underlying trends or relationships. Take the BEA data again: the *exact* values of the GMPs of all American cities in 2006 were to some extent the results of accidents, of chance, unrepeatable causes, and this would still be true even if we could measure those exact GMPs without error. To be really concrete for a moment, the city with the highest per-capita income in the data set is Bridgeport-Stamford-Norwalk, CT. This is a center for insurance companies, banks and hedge funds. Exactly how much money they made in 2006 depended on things like just how many hurricanes there were in Florida, wild fires in California, mortgages from Cleveland and Phoenix sold to small towns in Germany, etc., etc. Even if one thinks that there is, ultimately, some sort of deterministic explanation for all of these quantities, they're plainly very far removed from any general relationship between a city's population and its economic productivity. They really happen — they are not just measurement errors[4] — but they could easily have happened a bit differently. Long experience has shown that the *distribution* of these accidents is often stable, and can be modeled with probability theory[5]. When that happens, they are often called by the more respectable name of **fluctuations**.

To sum up: whether it's due to sampling, or measurement error, or fluctuations, we often have good reason to think that our data could have been more or less different. If we re-ran the experiment, or "re-wound the tape of history" (S. J. Gould), the results would not have been quite the same. This means that any statistic we calculate from the data would have been more or less different as well. When we try to quantify uncertainty, we want to know how different our calculations could have been.

To say anything useful here, we will need to make assumptions. Without *some* assumptions, we can't really say anything at all about how different the data could, plausibly, have been. In statistics, a lot of accumulated experience says that useful assumptions generally take the form of **statistical models** or **probability models**.

In a statistical model, we act as though the variables we measure, and possibly others we don't measure, are random variables. (We "model them as random variables.") The **specification** of a statistical model says what the random variables are, and lays down more or less narrow restrictions on their distributions and how they relate to each other. Here, for instance, are some *conceivable* statistical models for the BEA data. In all of them, X stands for a city's population and Y for its per-capita GMP.

1. $X \sim N(6.81\times 10^5, 2.42\times 10^{12})$; $Y|X \sim N(4.00\times 10^4, 8.50\times 10^7)$; X independent across cities; Y independent across cities given their X's.

2. $X \sim N(\mu_X, \sigma^2_X)$ for some mean μ_X and variance σ^2_X; $Y|X \sim N(4.00\times 10^4, 8.50\times 10^7)$; Y independent across cities given their X's.

3. $X \sim N(\mu_X, \sigma^2_X)$ for some mean μ_X and variance σ^2_X; $Y|X \sim N(4.00\times 10^4, \sigma^2_Y)$ for some variance σ^2_Y; Y independent across cities given their X's.

[4]"As I regularly find myself having to remind cadet risk managers with newly-minted PhDs in financial econometrics, the Great Depression did actually happen; it wasn't just a particularly innaccurate observation of the underlying 4% rate of return on equities." (`http://d-squareddigest.blogspot.com/2006/09/tail-events-phrase-considered-harmful.html`)

[5]In fact, there are precise mathematical senses in which sufficiently complicated deterministic processes end up looking just like random ones. If this intrigues you, see Ruelle (1991) and Smith (2007).

4. distribution of X unspecified; $Y|X \sim N(4.00 \times 10^4, \sigma_Y^2)$ for some σ_Y^2; Y independent across cities given their X's.

5. distribution of X unspecified; $Y|X \sim N(\beta_0 + \beta_1 x, \sigma_Y^2)$ for some $\beta_0, \beta_1, \sigma_Y^2$; Y independent across cities given their X's.

6. distribution of X unspecified; $\mathbb{E}[Y|X = x] = \beta_0 + \beta_1 x$ for some β_0 and β_1; $\mathrm{Var}[Y|X = x] = \sigma_Y^2$ for some σ_Y^2; Y independent across cities given their X's.

7. distribution of X unspecified; $\mathbb{E}[Y|X = x] = \beta_0 + \beta_1 x$ for some β_0 and β_1; Y uncorrelated across cities given their X's.

As we go down this list of models, we make weaker and weaker assumptions about the process which generated the data. This means that, with the same data, we can infer less and less about that data-generating process. The very first model specifies a single, complete, unambiguous distribution for the data. If we assumed that model was true, we could then make all sorts of assertions about the distribution of city population and economic output, without even having to look at the data at all. Later models in the list leave more about the data-generating process undetermined, so we can use the data to estimate those parts of the model (e.g., the mean and variance of city populations in model 2, or the slope β_1 in models from 4 on), or test hypotheses about them, etc. Because a model specifies how the data variable X and Y are distributed, and any statistics we calculate are functions of X and Y, a model also, implicitly, tells us how those statistics are distributed[6]. These distributions of the statistics are what we'll use to quantify the uncertainty in our inferences.

The stronger the assumptions we make, the stronger the inferences we can draw, *if the assumptions are true*. There is no virtue to strong conclusions which rest on faulty premises. Therefore: we are going to go over how to draw these inferences, but we are also going to go over checking model assumptions.

When confronting a data analysis problem, you first need to **formulate** a statistical model. This is partly about absorbing what's already known about the subject[7], partly about looking for similar problems others have dealt with and seeing what you can learn from them (i.e., analogy and tradition), and partly about being inspired by initial explorations of the data. Once you have a model, the two key tasks are **inference** within the model, and **checking** the model.

Inference within the model is about doing calculations which presume the model's assumptions are true: estimation, or prediction, or hypothesis testing, or confidence intervals, or what-have-you. This is usually what people actually want out of the data analysis. Unfortunately, these inferences are only as good as the modeling assumptions that they're based on.

Model checking, on the other hand, is about seeing whether the assumptions are really true. This includes formal goodness-of-fit testing, but also various "specification

[6]Whether we can work out that distribution in a nice closed form is another question, but mathematically exists, and there are ways to cope when there's no closed form, as we'll see later in this class, and in greater detail in 402.

[7]For instance, economists and geographers have long known about an "urban wage premium", where otherwise-similar workers in bigger cities get paid more (Thompson, 1968).

tests" (some of which we will cover), the less formal checks called "diagnostics", and sheer judgment, often aided by visualizations. If the assumptions of the model we started with turn out to be wrong, we need to go back and revise the model, either replacing the faulty assumptions with others, or weakening them.

People usually list put within-model inference before model checking — that's the order most textbooks use — but that's more because students, and teachers, are generally more comfortable with the more cut-and-dried topic of inference. That topic is extremely formalized and mathematical, with lots of theory to guide us. In fact, for lots of inference problems there is an unambiguous optimal procedure, which we should follow. Model checking, on the other hand, is much less formalized, mathematical and algorithmic than inference, and very little about it can be said to be definitely optimal or The Right Way To Do It. Nonetheless, *assumption checking is much more important*. Impressive-seeming inferences from strong-but-wrong assumptions don't actually tell us anything about the world, and are useless, no matter how much technical skill they might demonstrate. When reading other people's data analyses, you should get into the habit of paying very close attention to how they check their models, and you should apply that same habit to yourself.

2.3 The Simple Linear Regression Model

To make this philosophizing a bit more concrete, let's introduce the most basic of all statistical models that is actually useful for anything, the **simple linear regression model**. This is a model with two random variables, X and Y, where we are trying to predict Y from X. Here are the model's assumptions:

1. The distribution of X is unspecified, possibly even deterministic;
2. $Y|X = \beta_0 + \beta_1 x + \epsilon$, where ϵ is a noise variable;
3. ϵ has mean 0, a constant variance σ^2, and is uncorrelated with X and uncorrelated across observations.

The noise variable may represent measurement error, or fluctuations in Y, or some combination of both. The assumption of **additive** noise is non-trivial — it's not absurd to imagine that either measurement error or fluctuations might change Y multiplicatively (for instance). The assumption of a **linear functional form** for the relationship between Y and X is non-trivial; lots of non-linear relationships actually exist. The assumption of **constant variance**, or **homoskedasticity**, is non-trivial; the non-correlation assumptions are non-trivial. But the assumption that the noise has mean 0 *is* trivial. (Why?) Ideally, all of the non-trivial assumptions will be checked, and we will talk later in the course about ways to check them.

The assumptions I have just laid out, while they are non-trivial because they could be violated (and are, in many situations), are still strong enough to let us get a start on inference. While we will go into this in some detail next time, let's give this at least a start here.

Remember we saw last time that the optimal linear predictor of Y from X has slope $\beta_1 = \text{Cov}[X,Y]/\text{Var}[X]$. But both $\text{Cov}[X,Y]$ and $\text{Var}[X]$ are functions of the true distribution. Rather than having that full distribution, we merely have data points, say $(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)$. How might we *estimate* β_1 from this data?

An obvious approach would be to use the data to find the *sample* covariance and *sample* variance, and take their ratio. As a reminder, the sample variance of X is

$$s_X^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2 \tag{2.1}$$

while the sample covariance is

$$c_{XY} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y}) \tag{2.2}$$

(Here I am writing $\overline{x}$ for the sample average of the x_i, and similarly for other variables.)[8] So we'd have a **sample** (or **empirical**) slope

$$\widehat{\beta_1} = \frac{c_{XY}}{s_X^2} \tag{2.3}$$

We can't hope that $\widehat{\beta_1} = \beta_1$, but we *can* hope that as $n \to \infty$, $\widehat{\beta_1} \to \beta_1$. When an estimator converges on the truth like that, the estimator is called **consistent**, and this is the most basic property a good estimator should have. What do we need to assume in order for $\widehat{\beta_1} \to \beta_1$?

Let's look at the sample covariance. A little algebra shows

$$c_{XY} = \frac{1}{n}\sum_{i=1}^{n} x_i y_i - \bar{x}\bar{y} \tag{2.4}$$

According to the model, $y_i = (\beta_0 + \beta_1 x_i + \epsilon_i)$. So (after a little more algebra)

$$\begin{aligned} c_{XY} &= \frac{1}{n}\sum_{i=1}^{n} x_i(\beta_0 + \beta_1 x_i + \epsilon_i) - \overline{x}\overline{\beta_0 + \beta_1 x + \epsilon} && (2.5)\\ &= \beta_0\overline{x} + \beta_1\overline{x^2} + \overline{x\epsilon} - \overline{x}\beta_0 - \beta_1\overline{x}^2 - \bar{x}\bar{\epsilon} && (2.6)\\ &= \beta_1 s_X^2 + \overline{x\epsilon} - \bar{x}\bar{\epsilon} && (2.7)\end{aligned}$$

Because ϵ has mean 0, as $n \to \infty$, the law of large numbers says $\overline{\epsilon} \to 0$. Because ϵ is uncorrelated with x, using the law of large numbers again says that $\overline{x\epsilon} \to 0$ as well. So

$$c_{XY} \to \beta_1 s_X^2 \tag{2.8}$$

and therefore

$$\widehat{\beta_1} = \frac{c_{XY}}{s_X^2} \to \frac{\beta_1 s_X^2}{s_X^2} = \beta_1 \tag{2.9}$$

[8] Some people prefer to define these with denominators $n-1$ rather than n, to get unbiased estimates of the population quantities. The way I am doing it will simplify some book-keeping presently.

as desired.

As I said, this argument rests on all the model assumptions. Strictly speaking, the estimator $\widehat{\beta}$ is consistent under even weaker assumptions — it's enough that $c_{XY} \rightarrow \mathrm{Cov}[X,Y]$, and $s^2_X \rightarrow \mathrm{Var}[X]$. On the other hand, it would be nice to say more: we want to know *how far* from the truth our estimate is likely to be, whether it tends to over- or under- estimate the slope, etc. we will see in later chapters how the assumptions of the simple linear regression model will let us say something about all of these matters, and how the even stronger assumption that the noise is Gaussian will let us be even more precise.

(We will, I promise, come back to this data set, and the question of which regression line, if any, best describes the relationship between a city's size and its economic output, but that, too, will have to wait for later.)

Exercises

1. What, if anything, makes `plot(pcgmp ~ log(pop), data=bea)` a worse plot than `plot(pcgmp ~ pop, data=bea, log="x")`?
2. Fill in the algebra for (2.4).
3. Fill in the algebra for (2.5).

Chapter 3

Simple Linear Regression Models, with Hints at Their Estimation

3.1 The Simple Linear Regression Model

Let's recall the simple linear regression model from last time. This is a statistical model with two variables X and Y, where we try to predict Y from X. The assumptions of the model are as follows:

1. The distribution of X is arbitrary (and perhaps X is even non-random).
2. If $X = x$, then $Y = \beta_0 + \beta_1 x + \epsilon$, for some constants ("coefficients", "parameters") β_0 and β_1, and some random noise variable ϵ.
3. $\mathbb{E}[\epsilon|X = x] = 0$ (no matter what x is), $\mathrm{Var}[\epsilon|X = x] = \sigma^2$ (no matter what x is).
4. ϵ is uncorrelated across observations.

To elaborate, with multiple data points, $(X_1, Y_1), (X_2, Y_2), \ldots (X_n, Y_n)$, then the model says that, for *each* $i \in 1:n$,

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i \tag{3.1}$$

where the noise variables ϵ_i all have the same expectation (0) and the same variance (σ^2), and $\mathrm{Cov}\left[\epsilon_i, \epsilon_j\right] = 0$ (unless $i = j$, of course).

3.1.1 "Plug-In" Estimates

In Chapter 1, we saw that the optimal linear predictor of Y from X has slope $\beta_1 = \mathrm{Cov}[X, Y]/\mathrm{Var}[X]$, and intercept $\beta_0 = \mathbb{E}[Y] - \beta_1 \mathbb{E}[X]$. A common tactic in devising estimators is to use what's sometimes called the "plug-in principle", where we find equations for the parameters which would hold if we knew the full distribution, and "plug in" the sample versions of the population quantities. We saw this in Chapter 2, where we estimated β_1 by the ratio of the sample covariance to the sample variance:

$$\widehat{\beta_1} = \frac{c_{XY}}{s^2_X} \tag{3.2}$$

We also saw, in Chapter 2, that so long as the law of large numbers holds,

$$\widehat{\beta_1} \rightarrow \beta_1 \tag{3.3}$$

as $n \rightarrow \infty$. It follows easily that

$$\widehat{\beta_0} = \overline{Y} - \widehat{\beta_1}\overline{X} \tag{3.4}$$

will also converge on β_0.

3.1.2 Least Squares Estimates

An alternative way of estimating the simple linear regression model starts from the objective we are trying to reach, rather than from the formula for the slope. Recall, from Chapter 1, that the true optimal slope and intercept are the ones which minimize the mean squared error:

$$(\beta_0, \beta_1) = \underset{(b_0,b_1)}{\mathrm{argmin}} \mathbb{E}\left[(Y-(b_0+b_1X))^2\right] \tag{3.5}$$

This is a function of the complete distribution, so we can't get it from data, but we can approximate it with data. The **in-sample**, **empirical** or **training** MSE is

$$\widehat{MSE}(b_0, b_1) \equiv \frac{1}{n}\sum_{i=1}^{n}(y_i - (b_0 + b_1x_i))^2 \tag{3.6}$$

Notice that this is a function of b_0 and b_1; it is also, of course, a function of the data, $(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)$, but we will generally suppress that in our notation.

If our samples are all independent, for any fixed (b_0, b_1), the law of large numbers tells us that $\widehat{MSE}(b_0, b_1) \rightarrow MSE(b_0, b_1)$ as $n \rightarrow \infty$. So it doesn't seem unreasonable to try minimizing the in-sample error, which we can compute, as a proxy for minimizing the true MSE, which we can't. Where does it lead us?

Start by taking the derivatives with respect to the slope and the intercept:

$$\frac{\partial \widehat{MSE}}{\partial b_0} = \frac{1}{n}\sum_{i=1}^{n}(y_i - (b_0 + b_1x_i))(-2) \tag{3.7}$$

$$\frac{\partial \widehat{MSE}}{\partial b_1} = \frac{1}{n}\sum_{i=1}^{n}(y_i - (b_0 + b_1x_i))(-2x_i) \tag{3.8}$$

Set these to zero at the optimum $(\hat{\beta}_0, \hat{\beta}_1)$:

$$\begin{aligned} \frac{1}{n}\sum_{i=1}^{n}(y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i)) &= 0 \quad (3.9)\\ \frac{1}{n}\sum_{i=1}^{n}(y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i))(x_i) &= 0 \end{aligned}$$

These are often called the **normal equations** for least-squares estimation, or the **estimating equations**: a system of two equations in two unknowns, whose solution gives the estimate. Many people would, at this point, remove the factor of $1/n$, but I think it makes it easier to understand the next steps:

$$\begin{aligned} \overline{y} - \hat{\beta}_0 - \hat{\beta}_1 \overline{x} &= 0 \quad (3.10)\\ \overline{xy} - \hat{\beta}_0 \overline{x} - \hat{\beta}_1 \overline{x^2} &= 0 \quad (3.11) \end{aligned}$$

The first equation, re-written, gives

$$\hat{\beta}_0 = \overline{y} - \hat{\beta}_1 \overline{x} \quad (3.12)$$

Substituting this into the remaining equation,

$$\begin{aligned} 0 &= \overline{xy} - \bar{y}\bar{x} + \hat{\beta}_1 \bar{x}\bar{x} - \hat{\beta}_1 \overline{x^2} \quad (3.13)\\ 0 &= c_{XY} - \hat{\beta}_1 s_X^2 \quad (3.14)\\ \hat{\beta}_1 &= \frac{c_{XY}}{s_X^2} \quad (3.15) \end{aligned}$$

That is, the least-squares estimate of the slope is our old friend the plug-in estimate of the slope, and thus the least-squares intercept is also the plug-in intercept.

Going forward The equivalence between the plug-in estimator and the least-squares estimator is a bit of a special case for linear models. In some non-linear models, least squares is quite feasible (though the optimum can only be found numerically, not in closed form); in others, plug-in estimates are more useful than optimization.

3.1.3 Bias, Variance and Standard Error of Parameter Estimates

Whether we think of it as deriving from pluging-in or from least squares, we work out some of the properties of this estimator of the coefficients, using the model assumptions. We'll start with the slope, $\hat{\beta}_1$.

$$\begin{aligned}
\hat{\beta}_1 &= \frac{c_{XY}}{s^2_X} && (3.16)\\
&= \frac{\frac{1}{n}\sum_{i=1}^{n} x_i y_i - \bar{x}\bar{y}}{s^2_X} && (3.17)\\
&= \frac{\frac{1}{n}\sum_{i=1}^{n} x_i(\beta_0 + \beta_1 x_i + \epsilon_1) - \bar{x}(\beta_0 + \beta_1 \bar{x} + \bar{\epsilon})}{s^2_X} && (3.18)\\
&= \frac{\beta_0 \bar{x} + \beta_1 \overline{x^2} + \frac{1}{n}\sum_{i=1}^{n} x_i \epsilon_i - \bar{x}\beta_0 - \beta_1 \bar{x}^2 - \bar{x}\bar{\epsilon}}{s^2_x} && (3.19)\\
&= \frac{\beta_1 s^2_X + \frac{1}{n}\sum_{i=1}^{n} x_i \epsilon_i - \bar{x}\bar{\epsilon}}{s^2_X} && (3.20)\\
&= \beta_1 + \frac{\frac{1}{n}\sum_{i=1}^{n} x_i \epsilon_i - \bar{x}\bar{\epsilon}}{s^2_X} && (3.21)
\end{aligned}$$

Since $\bar{x}\bar{\epsilon} = n^{-1}\sum_i \bar{x}\epsilon_i$,

$$\hat{\beta}_1 = \beta_1 + \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})\epsilon_i}{s^2_X} \tag{3.22}$$

This representation of the slope estimate shows that it is equal to the true slope (β_1) plus something which depends on the noise terms (the ϵ_i, and their sample average $\bar{\epsilon}$). We'll use this to find the expected value and the variance of the estimator $\hat{\beta}_1$.

In the next couple of paragraphs, I am going to treat the x_i as non-random variables. This is appropriate in "designed" or "controlled" experiments, where we get to chose their value. In randomized experiments or in observational studies, obviously the x_i aren't necessarily fixed; however, these expressions will be correct for the conditional expectation $\mathbb{E}\left[\hat{\beta}_1 | x_1, \ldots x_n\right]$ and conditional variance $\mathrm{Var}\left[\hat{\beta}_1 | x_1, \ldots x_n\right]$, and I will come back to how we get the unconditional expectation and variance.

Expected value and bias Recall that $\mathbb{E}[\epsilon_i | X_i] = 0$, so

$$\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})\mathbb{E}[\epsilon_i] = 0 \tag{3.23}$$

Thus,

$$\mathbb{E}\left[\hat{\beta}_1\right] = \beta_1 \tag{3.24}$$

Since the **bias** of an estimator is the difference between its expected value and the truth, $\hat{\beta}_1$ is an **unbiased** estimator of the optimal slope.

(To repeat what I'm sure you remember from mathematical statistics: "bias" here is a technical term, meaning no more and no less than $\mathbb{E}\left[\hat{\beta}_1\right] - \beta_1$. An unbiased estimator could still make systematic mistakes — for instance, it could underestimate 99% of the time, provided that the 1% of the time it over-estimates, it does so by much more than it under-estimates. Moreover, unbiased estimators are not necessarily superior to biased ones: the total error depends on both the bias of the estimator and its variance, and there are many situations where you can remove lots of bias at the cost of adding a little variance. Least squares for simple linear regression happens not to be one of them, but you shouldn't expect that as a general rule.)

Turning to the intercept,

$$\begin{aligned}
\mathbb{E}\left[\hat{\beta}_0\right] &= \mathbb{E}\left[\overline{Y} - \hat{\beta}_1\overline{X}\right] && (3.25)\\
&= \beta_0 + \beta_1\overline{X} - \mathbb{E}\left[\hat{\beta}_1\right]\overline{X} && (3.26)\\
&= \beta_0 + \beta_1\overline{X} - \beta_1\overline{X} && (3.27)\\
&= \beta_0 && (3.28)
\end{aligned}$$

so it, too, is unbiased.

Variance and Standard Error Using the formula for the variance of a sum from Chapter 1, and the model assumption that all the ϵ_i are uncorrelated with each other,

$$\begin{aligned}
\mathrm{Var}\left[\hat{\beta}_1\right] &= \mathrm{Var}\left[\beta_1 + \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})\epsilon_i}{s_X^2}\right] && (3.29)\\
&= \mathrm{Var}\left[\frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})\epsilon_i}{s_X^2}\right] && (3.30)\\
&= \frac{\frac{1}{n^2}\sum_{i=1}^{n}(x_i - \bar{x})^2\mathrm{Var}\left[\epsilon_i\right]}{(s_X^2)^2} && (3.31)\\
&= \frac{\frac{\sigma^2}{n}s_X^2}{(s_X^2)^2} && (3.32)\\
&= \frac{\sigma^2}{n s_X^2} && (3.33)
\end{aligned}$$

In words, this says that the variance of the slope estimate goes up as the noise around the regression line (σ^2) gets bigger, and goes down as we have more observations (n), which are further spread out along the horizontal axis (s_X^2); it should not be surprising that it's easier to work out the slope of a line from many, well-separated points on the line than from a few points smushed together.

The **standard error** of an estimator is just its standard deviation, or the square root of its variance:

$$\mathrm{se}(\hat{\beta}_1) = \frac{\sigma}{\sqrt{n s_X^2}} \tag{3.34}$$

I will leave working out the variance of $\hat{\beta}_0$ as an exercise.

Unconditional-on-X Properties The last few paragraphs, as I said, have looked at the expectation and variance of $\hat{\beta}_1$ conditional on $x_1, \ldots x_n$, either because the x's really are non-random (e.g., controlled by us), or because we're just interested in conditional inference. If we do care about unconditional properties, then we still need to find $\mathbb{E}\left[\hat{\beta}_1\right]$ and $\text{Var}\left[\hat{\beta}_1\right]$, not just $\mathbb{E}\left[\hat{\beta}_1 | x_1, \ldots x_n\right]$ and $\text{Var}\left[\hat{\beta}_1 | x_1, \ldots x_n\right]$. Fortunately, this is easy, *so long as the simple linear regression model holds*.

To get the unconditional expectation, we use the "law of total expectation":

$$\begin{aligned} \mathbb{E}\left[\hat{\beta}_1\right] &= \mathbb{E}\left[\mathbb{E}\left[\hat{\beta}_1 | X_1, \ldots X_n\right]\right] & (3.35) \\ &= \mathbb{E}[\beta_1] = \beta_1 & (3.36) \end{aligned}$$

That is, the estimator is *unconditionally* unbiased.

To get the unconditional variance, we use the "law of total variance":

$$\begin{aligned} \text{Var}\left[\hat{\beta}_1\right] &= \mathbb{E}\left[\text{Var}\left[\hat{\beta}_1 | X_1, \ldots X_n\right]\right] + \text{Var}\left[\mathbb{E}\left[\hat{\beta}_1 | X_1, \ldots X_n\right]\right] & (3.37) \\ &= \mathbb{E}\left[\frac{\sigma^2}{n s_X^2}\right] + \text{Var}[\beta_1] & (3.38) \\ &= \frac{\sigma^2}{n}\mathbb{E}\left[\frac{1}{s_X^2}\right] & (3.39) \end{aligned}$$

3.1.4 Parameter Interpretation; Causality

Two of the parameters are easy to interpret.

σ^2 is the variance of the noise around the regression line; σ is a typical distance of a point from the line. ("Typical" here in a special sense, it's the root-mean-squared distance, rather than, say, the average absolute distance.)

β_0 is the simply the expected value of Y when X is 0, $\mathbb{E}[Y|X=0]$. The point $X=0$ usually has no special significance, but this setting does ensure that the line goes through the point $(\mathbb{E}[X], \mathbb{E}[Y])$.

The interpretation of the slope is both very straightforward and very tricky. Mathematically, it's easy to convince yourself that, for any x

$$\beta_1 = \mathbb{E}[Y|X=x] - \mathbb{E}[Y|X=x-1] \tag{3.40}$$

or, for any x_1, x_2,

$$\beta_1 = \frac{\mathbb{E}[Y|X=x_2] - \mathbb{E}[Y|X=x_1]}{x_2 - x_1} \tag{3.41}$$

This is just saying that the slope of a line is "rise/run".

The tricky part is that we have a *very* strong, natural tendency to interpret this as telling us something about causation — "If we change X by 1, then on average Y will change by β_1". This interpretation is usually completely unsupported by the analysis. If I use an old-fashioned mercury thermometer, the height of mercury in the tube usually has a nice linear relationship with the temperature of the room the thermometer is in. This linear relationship goes both ways, so we could regress temperature (Y) on mercury height (X). But if I manipulate the height of the mercury (say, by changing the ambient pressure, or shining a laser into the tube, etc.), changing the height X will not, in fact, change the temperature outside.

The right way to interpret β_1 is not as the result of a *change*, but as an expected *difference*. The correct catch-phrase would be something like "If we select two sets of cases from the un-manipulated distribution where X differs by 1, we expect Y to differ by β_1." This covers the thermometer example, and every other I can think of. It is, I admit, much more inelegant than "If X changes by 1, Y changes by β_1 on average", but it has the advantage of being true, which the other does not.

There *are* circumstances where regression can be a useful part of causal inference, but we will need a lot more tools to grasp them; that will come towards the end of 402.

3.2 The Gaussian-Noise Simple Linear Regression Model

We have, so far, assumed comparatively little about the noise term ϵ. The advantage of this is that our conclusions apply to lots of different situations; the drawback is that there's really not all that much more to say about our estimator $\widehat{\beta}$ or our predictions than we've already gone over. If we made more detailed assumptions about ϵ, we could make more precise inferences.

There are lots of forms of distributions for ϵ which we might contemplate, and which are compatible with the assumptions of the simple linear regression model (Figure 3.1). The one which has become the most common over the last two centuries is to assume ϵ follows a Gaussian distribution.

The result is the Gaussian-noise simple linear regression model[1]:

1. The distribution of X is arbitrary (and perhaps X is even non-random).

2. If $X = x$, then $Y = \beta_0 + \beta_1 x + \epsilon$, for some constants ("coefficients", "parameters") β_0 and β_1, and some random noise variable ϵ.

3. $\epsilon \sim N(0, \sigma^2)$, independent of X.

4. ϵ is independent across observations.

[1]Our textbook, rather old-fashionedly, calls this the "normal error" model rather than "Gaussian noise". I dislike this: "normal" is an over-loaded word in math, while "Gaussian" is (comparatively) specific; "error" made sense in Gauss's original context of modeling, specifically, errors of observation, but is misleading generally; and calling Gaussian distributions "normal" suggests they are much more common than they really are.

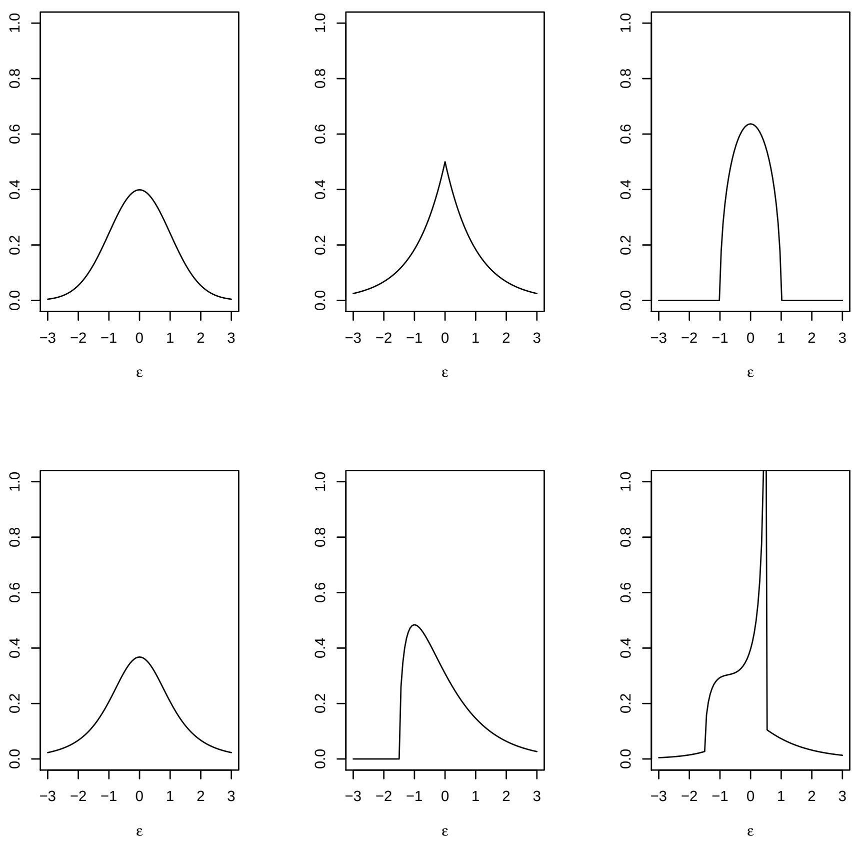


FIGURE 3.1: *Some possible noise distributions for the simple linear regression model, since all have* $\mathbb{E}[\epsilon] = 0$*, and could get any variance by scaling. (The model is even compatible with each observation taking* ϵ *from a different distribution.) From top left to bottom right: Gaussian; double-exponential ("Laplacian"); "circular" distribution; t with 3 degrees of freedom; a gamma distribution (shape 1.5, scale 1) shifted to have mean 0; mixture of two gammas with shape 1.5 and shape 0.5, each off-set to have expectation 0. The first three were all used as error models in the 18th and 19th centuries. (See Figure 3.2 for the code.)*

```
par(mfrow = c(2, 3))
curve(dnorm(x), from = -3, to = 3, xlab = expression(epsilon), ylab = "", ylim = c(0,
    1))
curve(exp(-abs(x))/2, from = -3, to = 3, xlab = expression(epsilon), ylab = "",
    ylim = c(0, 1))
curve(sqrt(pmax(0, 1 - x^2))/(pi/2), from = -3, to = 3, xlab = expression(epsilon),
    ylab = "", ylim = c(0, 1))
curve(dt(x, 3), from = -3, to = 3, xlab = expression(epsilon), ylab = "", ylim = c(0,
    1))
curve(dgamma(x + 1.5, shape = 1.5, scale = 1), from = -3, to = 3, xlab = expression(epsilon),
    ylab = "", ylim = c(0, 1))
curve(0.5 * dgamma(x + 1.5, shape = 1.5, scale = 1) + 0.5 * dgamma(0.5 - x,
    shape = 0.5, scale = 1), from = -3, to = 3, xlab = expression(epsilon),
    ylab = "", ylim = c(0, 1))
par(mfrow = c(1, 1))
```

FIGURE 3.2: *Code for producing Figure 3.1.*

You will notice that these assumptions are strictly stronger than those of the simple linear regression model. More exactly, the first two assumptions are the same, while the third and fourth assumptions of the Gaussian-noise model imply the corresponding assumptions of the other model. This means that everything we have done so far directly applies to the Gaussian-noise model. On the other hand, the stronger assumptions let us say more. They tell us, exactly, the probability distribution for Y given X, and so will let us get exact distributions for predictions and for other inferential statistics.

Why the Gaussian noise model? Why should we think that the noise around the regression line would follow a Gaussian distribution, independent of X? There are two big reasons.

1. *Central limit theorem* The noise might be due to adding up the effects of lots of little random causes, all nearly independent of each other and of X, where each of the effects are of roughly similar magnitude. Then the central limit theorem will take over, and the distribution of the sum of effects will indeed be pretty Gaussian. For Gauss's original context, X was (simplifying) "Where is such-and-such-a-planet in space?", Y was "Where does an astronomer record the planet as appearing in the sky?", and noise came from defects in the telescope, eye-twitches, atmospheric distortions, etc., etc., so this was pretty reasonable. It is clearly *not* a universal truth of nature, however, or even something we should expect to hold true as a general rule, as the name "normal" suggests.

2. *Mathematical convenience* Assuming Gaussian noise lets us work out a very complete theory of inference and prediction for the model, with lots of closed-form answers to questions like "What is the optimal estimate of the variance?" or "What is the probability that we'd see a fit this good from a line with a non-

zero intercept if the true line goes through the origin?", etc., etc. Answering such questions without the Gaussian-noise assumption needs somewhat more advanced techniques, and much more advanced computing; we'll get to it towards the end of the class.

3.2.1 Visualizing the Gaussian Noise Model

The Gaussian noise model gives us not just an expected value for Y at each x, but a whole conditional distribution for Y at each x. To visualize it, then, it's not enough to just sketch a curve; we need a three-dimensional surface, showing, for each combination of x and y, the probability density of Y around that y given that x. Figure 3.3 illustrates.

3.2.2 Maximum Likelihood vs. Least Squares

As you remember from your mathematical statistics class, the **likelihood** of a parameter value on a data set is the probability density at the data under those parameters. We could not work with the likelihood with the simple linear regression model, because it didn't specify enough about the distribution to let us calculate a density. With the Gaussian-noise model, however, we can write down a likelihood[2] By the model's assumptions, if think the parameters are the parameters are b_0, b_1, s^2 (reserving the Greek letters for their true values), then $Y|X = x \sim N(b_0 + b_1 x, s^2)$, and Y_i and Y_j are independent given X_i and X_j, so the over-all likelihood is

$$\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi s^2}} e^{-\frac{(y_i-(b_0+b_1 x_i))^2}{2s^2}} \tag{3.42}$$

As usual, we work with the log-likelihood, which gives us the same information[3] but replaces products with sums:

$$L(b_0, b_1, s^2) = -\frac{n}{2}\log 2\pi - \frac{n}{\log} s - \frac{1}{2s^2}\sum_{i=1}^{n}(y_i - (b_0 + b_1 x_i))^2 \tag{3.43}$$

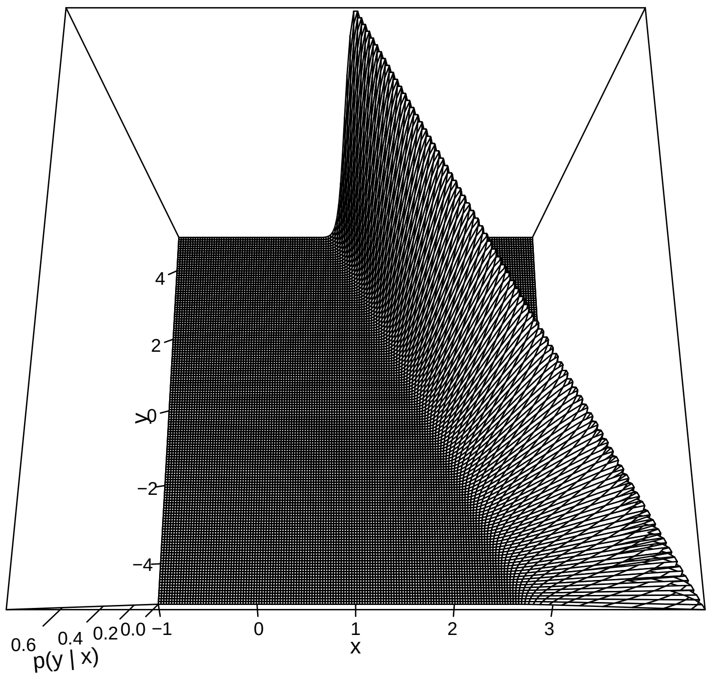
We recall from mathematical statistics that when we've got a likelihood function, we generally want to maximize it. That is, we want to find the parameter values which make the data we observed as likely, as probable, as the model will allow. (This may not be very likely; that's another issue.) We recall from calculus that one way to maximize is to take derivatives and set them to zero.

$$\frac{\partial L}{\partial b_0} = -\frac{1}{2s^2}\sum_{i=1}^{n} 2(y_i - (b_0 + b_1 x_i))(-1) \tag{3.44}$$

$$\frac{\partial L}{\partial b_1} = -\frac{1}{2s^2}\sum_{i=1}^{n} 2(y_i - (b_0 + b_1 x_i))(-x_i) \tag{3.45}$$

[2]Strictly speaking, this is a "conditional" (on X) likelihood; but only pedants use the adjective in this context.

[3]Why is this?



```
x.seq <- seq(from = -1, to = 3, length.out = 150)
y.seq <- seq(from = -5, to = 5, length.out = 150)
cond.pdf <- function(x, y) {
    dnorm(y, mean = 10 - 5 * x, sd = 0.5)
}
z <- outer(x.seq, y.seq, cond.pdf)
persp(x.seq, y.seq, z, ticktype = "detailed", phi = 75, xlab = "x", ylab = "y",
    zlab = expression(p(y | x)), cex.axis = 0.8)
```

FIGURE 3.3: *Illustrating how the conditional pdf of Y varies as a function of X, for a hypothetical Gaussian noise simple linear regression where $\beta_0 = 10$, $\beta_1 = -5$, and $\sigma^2 = (0.5)^2$. The perspective is adjusted so that we are looking nearly straight down from above on the surface. (Can you find a better viewing angle?) See* `help(persp)` *for the 3D plotting (especially the examples), and* `help(outer)` *for the* `outer` *function, which takes all combinations of elements from two vectors and pushes them through a function. How would you modify this so that the regression line went through the origin with a slope of 4/3 and a standard deviation of 5?*

Notice that when we set these derivatives to zero, all the multiplicative constants — in particular, the prefactor of $\frac{1}{2s^2}$ — go away. We are left with

$$\sum_{i=1}^{n} y_i - (\widehat{\beta_0} + \widehat{\beta_1} x_i) = 0 \tag{3.46}$$

$$\sum_{i=1}^{n} (y_i - (\widehat{\beta_0} + \widehat{\beta_1} x_i)) x_i = 0 \tag{3.47}$$

These are, up to a factor of $1/n$, *exactly* the equations we got from the method of least squares (Eq. 3.9). That means that the least squares solution *is* the maximum likelihood estimate under the Gaussian noise model; this is no coincidence[4].

Now let's take the derivative with respect to s:

$$\frac{\partial L}{\partial s} = -\frac{n}{s} + 2\frac{1}{2s^3}\sum_{i=1}^{n}(y_i - (b_0 + b_1 x_i))^2 \tag{3.48}$$

Setting this to 0 at the optimum, including multiplying through by $\widehat{\sigma}^3$, we get

$$\widehat{\sigma^2} = \frac{1}{n}\sum_{i=1}^{n}(y_i - (\widehat{\beta_0} + \widehat{\beta_1} x_i))^2 \tag{3.49}$$

Notice that the right-hand side is just the in-sample mean squared error.

Other models Maximum likelihood estimates of the regression curve coincide with least-squares estimates when the noise around the curve is additive, Gaussian, of constant variance, and both independent of X and of other noise terms. These were all assumptions we used in setting up a log-likelihood which was, up to constants, proportional to the (negative) mean-squared error. If any of those assumptions fail, maximum likelihood and least squares estimates can diverge, though sometimes the MLE solves a "generalized" least squares problem (as we'll see later in this course).

Exercises

1. Show that if $\mathbb{E}[\epsilon | X = x] = 0$ for all x, then $\mathrm{Cov}[X, \epsilon] = 0$. Would this still be true if $\mathbb{E}[\epsilon | X = x] = a$ for some other constant a?

2. Find the variance of $\hat{\beta}_0$. *Hint:* Do you need to worry about covariance between $\overline{Y}$ and $\hat{\beta}_1$?

[4]It's no coincidence because, to put it somewhat anachronistically, what Gauss did was ask himself "for what distribution of the noise would least squares maximize the likelihood?".

Chapter 4

The Method of Least Squares for Simple Linear Regression

Contents

4.1 Recapitulation

Let's recap from last time. The simple linear regression model is a statistical model for two variables, X and Y. We use X — the **predictor** variable — to try to predict Y, the **target** or **response**[1]. The assumptions of the model are:

1. The distribution of X is arbitrary (and perhaps X is even non-random).
2. If $X = x$, then $Y = \beta_0 + \beta_1 x + \epsilon$, for some constants ("coefficients", "parameters") β_0 and β_1, and some random noise variable ϵ.
3. $\mathbb{E}[\epsilon|X = x] = 0$ (no matter what x is), $\mathrm{Var}[\epsilon|X = x] = \sigma^2$ (no matter what x is).
4. ϵ is uncorrelated across observations.

In a typical situation, we also possess observations $(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)$, which we presume are a realization of the model. Our goals are to estimate the parameters of the model, and to use those parameters to make predictions.

In Chapter 3, we saw that we could estimate the parameters by the **method of least squares**: that is, of minimizing the in-sample mean squared error:

$$\widehat{MSE}(b_0, b_1) \equiv \frac{1}{n}\sum_{i=1}^{n}(y_i - (b_0 + b_1 x_i))^2 \tag{4.1}$$

In particular, we obtained the following results:

Normal or estimating equations The least-squares estimates solve the **normal** or **estimating** equations:

$$\begin{aligned} \overline{y} - \hat{\beta}_0 - \hat{\beta}_1\overline{x} &= 0 \quad (4.2)\\ \overline{xy} - \hat{\beta}_0\overline{x} - \hat{\beta}_1\overline{x^2} &= 0 \quad (4.3) \end{aligned}$$

[1]Older terms would be "independent" and "dependent" variables, respectively. These import an unwarranted suggestion of causality or even deliberate manipulation on the part of X, so I will try to avoid them.

Closed-form solutions The solution to the estimating equations can be given in closed form:

$$\hat{\beta}_1 = \frac{c_{XY}}{s_X^2} \tag{4.4}$$

$$\hat{\beta}_0 = \overline{y} - \hat{\beta}_1 \overline{x} \tag{4.5}$$

Unbiasedness The least-squares estimator is unbiased:

$$\mathbb{E}\left[\hat{\beta}_0\right] = \beta_0 \tag{4.6}$$

$$\mathbb{E}\left[\hat{\beta}_1\right] = \beta_1 \tag{4.7}$$

Variance shrinks like $1/n$ The variance of the estimator goes to 0 as $n \rightarrow \infty$, like $1/n$:

$$\mathrm{Var}\left[\hat{\beta}_1\right] = \frac{\sigma^2}{n s_X^2} \tag{4.8}$$

$$\mathrm{Var}\left[\hat{\beta}_0\right] = \frac{\sigma^2}{n}\left(1 + \frac{\overline{x}^2}{s_X^2}\right) \tag{4.9}$$

In these notes, I will try to explain a bit more of the general picture underlying these results, and to explain what it has to do with prediction.

4.2 In-Sample MSE vs. True MSE

The true regression coefficients minimize the true MSE, which is (under the simple linear regression model):

$$(\beta_0, \beta_1) = \underset{(b_0,b_1)}{\operatorname{argmin}} \mathbb{E}\left[(Y - (b_0 + b_1 X))^2\right] \tag{4.10}$$

What we minimize instead is the mean squared error on the data:

$$(\hat{\beta}_0, \hat{\beta}_1) = \underset{(b_0,b_1)}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n}(y_i - (b_0 + b_1 x_i))^2 \tag{4.11}$$

This is the **in-sample** or **empirical** version of the MSE. It's clear that it's a sample average, so for any *fixed* parameters b_0, b_1, when the law of large numbers applies, we should have

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - (b_0 + b_1 x_i))^2 \rightarrow \mathbb{E}\left[(Y - (b_0 + b_1 X))^2\right] \tag{4.12}$$

as $n \rightarrow \infty$. This should make it *plausible* that the minimum of the function of the left is going to converge on the minimum of the function on the right, but there can be tricky situations, with more complex models, where this convergence doesn't happen.

To illustrate what I mean by this convergence, Figure 4.2 shows a sequence of surfaces of the MSE as a function of (b_0, b_1). (The simulation code is in Figure 4.1.) The first row shows different in-sample MSE surfaces at a small value of n; the next row at a larger value of n; the next row at a still larger value of n. What you can see is that as n grows, these surfaces all become more similar to each other, and the locations of the minima are also becoming more similar. This isn't a *proof*, but shows why it's worth looking for a proof.

4.2.1 Existence and Uniqueness

On any given finite data set, it is evident from Eqs. 4.4–4.5 that there is always a least-squares estimate, *unless* $s_X^2 = 0$, i.e., unless the sample variance of X is zero, i.e., unless all the x_i have the same value. (Obviously, with only one value of the x coordinate, we can't work out the slope of a line!) Moreover, if $s_X^2 > 0$, then there is exactly *one* combination of slope and intercept which minimizes the MSE in-sample.

One way to understand this algebraically is that the estimating equations give us a system of two linear equations in two unknowns. As we remember from linear algebra (or earlier), such systems have a unique solution, unless one of the equations of the system is redundant. (See Exercise 2.)

Notice that this existence and uniqueness of a least-squares estimate assumes *absolutely nothing* about the data-generating process. In particular, it does *not* assume that the simple linear regression model is correct. There is always *some* straight line that comes closest to our data points, no matter how wrong, inappropriate or even just plain silly the simple linear model might be.

4.3 Constant-Plus-Noise Representations

In deriving the properties of the least-squares estimators, it is extremely helpful to re-write them so that they have the form "constant + noise", and especially to try to write the noise as a sum of uncorrelated random variables. This sort of "representation" of the estimator makes it much simpler to determine its properties, because adding up constants and uncorrelated random variables is what the rules of algebra from Chapter 1 make easy for us.

To this end, let's be explicit about writing out $\hat{\beta}_1$ in the form of a constant plus a sum of uncorrelated noise random variables.

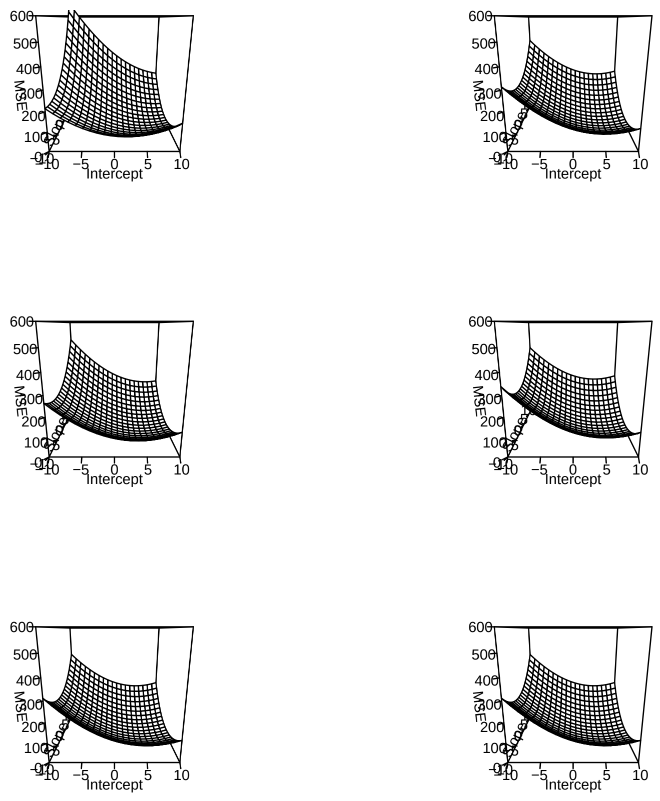
Begin with the fact that $\hat{\beta}_1$ is the ratio of the sample covariance to the sample variance of X:

$$\begin{aligned}
\hat{\beta}_1 &= \frac{c_{XY}}{s_X^2} && (4.13)\\
&= \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{s_X^2} && (4.14)\\
&= \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})y_i - \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})\overline{y}}{s_X^2} && (4.15)
\end{aligned}$$

```
# Simulate from a linear model with uniform X and t-distributed noise
# Inputs: number of points; intercept; slope; width of uniform X
# distribution (symmetric around 0); degrees of freedom for t Output: data
# frame with columns for X and Y
sim.linmod <- function(n, beta.0, beta.1, width, df) {
    # draw n points from a uniform distribution centered on 0
    x <- runif(n, min = -width/2, max = width/2)
    # draw n points from a t distribution with the given number of degrees of
    # freedom
    epsilon <- rt(n, df = df)
    # make y from a linear model
    y <- beta.0 + beta.1 * x + epsilon
    # return the data frame
    return(data.frame(x = x, y = y))
}

# Calculate in-sample MSE of a linear model First define a function that
# works for just one slope/intercept pair at time Then 'Vectorize' it to
# handle vectors of intercepts and slopes Inputs: slope; intercept; data
# frame with 'x' and 'y' columns Output: the in-sample MSE Presumes: 'y' is
# the target variable and 'x' is the predictor
mse.insample <- function(b.0, b.1, data) {
    mean((data$y - (b.0 + b.1 * data$x))^2)
}
mse.insample <- Vectorize(mse.insample, vectorize.args = c("b.0", "b.1"))

# Grids of possible intercepts and slopes
b.0.seq <- seq(from = -10, to = 10, length.out = 20)
b.1.seq <- seq(from = -10, to = 10, length.out = 20)

# 3d wire-mesh ('perspective') plot of a linear model's error surface Input:
# data set; maximum value for Z axis (for comparability across plots)
# Output: Transformation matrix for adding new points/lines to the plot,
# invisibly --- see help(persp) under 'Value'.  (Ignored here) ATTN:
# hard-coded slope/intercept sequences less than ideal
in.sample.persp <- function(data, zmax = 600) {
    # Calculate the in-sample MSE for every combination of
    z <- outer(b.0.seq, b.1.seq, mse.insample, data = data)
    persp(b.0.seq, b.1.seq, z, zlim = c(0, zmax), xlab = "Intercept", ylab = "Slope",
        zlab = "MSE", ticktype = "detailed")
}
```

FIGURE 4.1: *Code to simulate from a linear model with t-distributed noise and uniformly distributed X (to emphasize here needs anything to be Gaussian); to calculate the MSE of a linear model on a given data sample; and to plot the error surface on a given data set.*

```
## Warning in persp.default(b.0.seq, b.1.seq, z, zlim = c(0, zmax), xlab =
"Intercept", :  surface extends beyond the box
```



```
par(mfrow = c(3, 2))
in.sample.persp(sim.linmod(n = 10, beta.0 = 5, beta.1 = -2, width = 4, df = 3))
in.sample.persp(sim.linmod(n = 10, beta.0 = 5, beta.1 = -2, width = 4, df = 3))
in.sample.persp(sim.linmod(n = 100, beta.0 = 5, beta.1 = -2, width = 4, df = 3))
in.sample.persp(sim.linmod(n = 100, beta.0 = 5, beta.1 = -2, width = 4, df = 3))
in.sample.persp(sim.linmod(n = 1e+05, beta.0 = 5, beta.1 = -2, width = 4, df = 3))
in.sample.persp(sim.linmod(n = 1e+05, beta.0 = 5, beta.1 = -2, width = 4, df = 3))
par(mfrow = c(1, 1))
```

FIGURE 4.2: *Error surfaces for the linear model $Y = 5 - 2X + \epsilon$, $\epsilon \sim t_3$, $X \sim U(-2, 2)$, at $n = 10$ (top row), $n = 100$ (middle) and $n = 100000$ (bottom). Each column is an independent run of the model. Notice how these become increasingly similar as n grows.*

At this point, we need to pause for a fundamental fact which we will use often: for *any* variable z, the average difference from the sample mean is zero: $n^{-1}\sum_i z_i - \overline{z} = 0$. To see this, break up the sum of the difference into a difference in sums:

$$\frac{1}{n}\sum_{i=1}^{n} z_i - \overline{z} = \frac{1}{n}\sum_{i=1}^{n} z_i - \frac{1}{n}\sum_{i=1}^{n}\overline{z} \tag{4.16}$$

$$= \overline{z} - \frac{n\overline{z}}{n} = 0 \tag{4.17}$$

It follows that for any w which is constant in i,

$$\frac{1}{n}\sum_{i=1}^{n}(z_i - \overline{z})w = 0 \tag{4.18}$$

Thus

$$\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})\overline{y} = 0 \tag{4.19}$$

So

$$\hat{\beta}_1 = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})y_i}{s_X^2} \tag{4.20}$$

So far, we have not used any of our modeling assumptions. We now do so. Specifically, we use the assumption that

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i \tag{4.21}$$

For reasons which should become clear momentarily, it will be more convenient to write this in terms of how far x_i is from the sample mean $\overline{x}$:

$$y_i = \beta_0 + \beta_1\overline{x} + \beta_1(x_i - \overline{x}) + \epsilon_i \tag{4.22}$$

to substitute the above expression for y_i into Eq. 4.20:

$$\hat{\beta}_1 = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})(\beta_0 + \beta_1\overline{x} + \beta_1(x_i - \overline{x}) + \epsilon_i)}{s_X^2} \tag{4.23}$$

$$= \frac{\frac{\beta_0+\beta_1\overline{x}}{n}\sum_{i=1}^{n}(x_i - \overline{x}) + \frac{\beta_1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2 + \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})\epsilon_i}{s_X^2} \tag{4.24}$$

The first sum in the numerator is a constant times the average difference of x_i from $\overline{x}$, so it's zero (by Eq. 4.18). The second sum in the numerator is just s_X^2 again. In the third sum, because the ϵ_i are *not* constant, is not (necessarily) zero. Simplifying:

$$\hat{\beta}_1 = \beta_1 + \sum_{i=1}^{n}\frac{x_i - \overline{x}}{n s_X^2}\epsilon_i \tag{4.25}$$

Notice the form of Eq. 4.25: it writes our random estimator as a constant plus a weighted sum of the noise terms ϵ_i. In fact, by the fourth item in our listing of assumptions for the simple linear regression model, it writes $\hat{\beta}_1$ as a constant plus a weighted sum of *uncorrelated* noise terms.

It is now very easy to work out the expected value:

$$\begin{aligned} \mathbb{E}\left[\hat{\beta}_1\right] &= \mathbb{E}\left[\beta_1 + \sum_{i=1}^{n} \frac{x_i - \overline{x}}{n s_X^2} \epsilon_i\right] && (4.26) \\ &= \beta_1 + \sum_{i=1}^{n} \frac{x_i - \overline{x}}{s_X^2} \mathbb{E}[\epsilon_i] = \beta_1 && (4.27) \end{aligned}$$

or the variance:

$$\begin{aligned} \mathrm{Var}\left[\hat{\beta}_1\right] &= \mathrm{Var}\left[\beta_1 + \sum_{i=1}^{n} \frac{x_i - \overline{x}}{n s_X^2} \epsilon_i\right] && (4.28) \\ &= \mathrm{Var}\left[\sum_{i=1}^{n} \frac{x_i - \overline{x}}{n s_X^2} \epsilon_i\right] && (4.29) \\ &= \sum_{i=1}^{n} \frac{(x_i - \overline{x})^2}{n^2 s_X^4} \mathrm{Var}[\epsilon_i] && (4.30) \\ &= \sigma^2 \frac{n s_X^2}{n^2 s_X^4} = \frac{\sigma^2}{n s_X^2} && (4.31) \end{aligned}$$

where the last line uses the modeling assumption that all of the ϵ_i have the same variance. (The next-to-last line uses the assumption that they are uncorrelated.)

So far, this is just re-capitulating stuff we've done already, but the exact same strategy works for any estimator (or test statistic, etc.) which we can manipulate into constant-plus-noise form. It's not *always* possible to do this (though see the optional section 4.9, and, for the ambitious, van der Vaart 1998), but it's a very powerful strategy when it works. To illustrate its power, we'll now use it on predictions of the simple linear model, when estimated by least squares.

4.4 Predictions

Remember that we got into all this mess not because we want to know the numbers β_0 and β_1 for their own sake, but because we wanted to predict Y from X. How do we make those predictions, and how good are they?

If we knew β_0 and β_1, and that $X = x$, then our prediction[2] for Y would be $\beta_0 + \beta_1 x$. This is, assuming the simple linear regression model is true, exactly $\mathbb{E}[Y|X = x]$, which we saw back in Chapter 1 is the best prediction we can make. As x changes,

[2]This is called a **point** prediction; think of it as "if you have to give one number, this is the best single number to give." We might also make **interval** predictions (e.g., "with probability p, Y will be in the interval $[l, u]$") or **distributional** predictions (e.g., "Y will follow an $N(m,, v)$ distribution".

this prediction changes, but that's precisely what we want — the predictions will just follow points on the line.

Since we do not know β_0 and β_1, we fake it — that is, we use our estimates of the coefficients. At an *arbitrary* value of X, say x (sometimes called the **operating point**), we predict that on average Y will be

$$\hat{m}(x) = \hat{\beta}_0 + \hat{\beta}_1 x \tag{4.32}$$

This point prediction is called the **fitted value**[3] at x.

Notice the fitted value $\hat{m}(x)$ is an *estimate* of $\mathbb{E}[Y|X=x]$. The latter is a perfectly deterministic quantity; it has the value $\beta_0+\beta_1 x$, which is some number or other, and we just happen not to know it. But $\hat{m}(x)$ is a function of our data, which are random, hence $\hat{m}(x)$ is also random. It inherits its randomness from $\hat{\beta}_0$ and $\hat{\beta}_1$, which in turn inherit theirs from $\overline{y}$ and c_{XY}.

To analyze the randomness in $\hat{m}(x)$, we will represent it as constants plus a weighted sum of uncorrelated noise terms. Using Eqs. 4.5,

$$\begin{aligned} \hat{m}(x) &= \hat{\beta}_0 + \hat{\beta}_1 x && (4.33)\\ &= \overline{y} - \hat{\beta}_1 \overline{x} + \hat{\beta}_1 x && (4.34)\\ &= \overline{y} + (x-\overline{x})\hat{\beta}_1 && (4.35)\end{aligned}$$

Using Eq. 4.25 and the definition of a sample mean,

$$\begin{aligned} \hat{m}(x) &= \frac{1}{n}\sum_{i=1}^{n} y_i + (x-\overline{x})\left(\beta_1 + \sum_{i=1}^{n}\frac{x_i-\overline{x}}{ns_X^2}\epsilon_i\right) && (4.36)\\ &= \frac{1}{n}\sum_{i=1}^{n}(\beta_0+\beta_1 x_i+\epsilon_i) + (x-\overline{x})\left(\beta_1 + \sum_{i=1}^{n}\frac{x_i-\overline{x}}{ns_X^2}\epsilon_i\right) && (4.37)\\ &= \beta_0 + \beta_1\overline{x} + \frac{1}{n}\sum_{i=1}^{n}\epsilon_i + (x-\overline{x})\beta_1 + (x-\overline{x})\sum_{i=1}^{n}\frac{x_i-\overline{x}}{ns_X^2}\epsilon_i && (4.38)\\ &= \beta_0 + \beta_1 x + \frac{1}{n}\sum_{i=1}^{n}\left(1+(x-\overline{x})\frac{x_i-\overline{x}}{s_X^2}\right)\epsilon_i && (4.39)\end{aligned}$$

where in the last line I've canceled $\beta_1\overline{x}$ terms of opposite sign, and combined terms in the ϵ_i. Also, the second line used the second assumption in the simple linear regression model, that Y is a linear function of X plus noise.

Now we can check whether or not our predictions are biased:

$$\begin{aligned} \mathbb{E}[\hat{m}(x)] &= \mathbb{E}\left[\beta_0 + \beta_1 x + \frac{1}{n}\sum_{i=1}^{n}\left(1+(x-\overline{x})\frac{x_i-\overline{x}}{s_X^2}\right)\epsilon_i\right] && (4.40)\\ &= \beta_0 + \beta_1 x + \frac{1}{n}\sum_{i=1}^{n}\left(1+(x-\overline{x})\frac{x_i-\overline{x}}{s_x^2}\right)\mathbb{E}[\epsilon_i] && (4.41)\\ &= \beta_0 + \beta_1 x && (4.42)\end{aligned}$$

[3]The name originates from thinking of ϵ as purely measurement error, so that $\hat{m}(x)$ is our best-fitting estimate of the true value at x.

This is to say, no, under the simple linear model, the predictions of least squares are unbiased.

Of course, our predictions are somewhat random, because (as I said) they're functions of the somewhat-random data we estimated the model on. What is the variance of these predictions?

$$\begin{aligned}
\text{Var}[\hat{m}(x)] &= \text{Var}\left[\beta_0 + \beta_1 x + \frac{1}{n}\sum_{i=1}^{n}\left(1 + (x-\overline{x})\frac{x_i - \overline{x}}{s_X^2}\right)\epsilon_i\right] && (4.43)\\
&= \text{Var}\left[\frac{1}{n}\sum_{i=1}^{n}\left(1 + (x-\overline{x})\frac{x_i - \overline{x}}{s_X^2}\right)\epsilon_i\right] && (4.44)\\
&= \frac{1}{n^2}\sum_{i=1}^{n}\left(1 + (x-\overline{x})\frac{x_i - \overline{x}}{s_X^2}\right)^2 \text{Var}[\epsilon_i] && (4.45)\\
&= \frac{\sigma^2}{n^2}\sum_{i=1}^{n} 1 + 2(x-\overline{x})\frac{x_i - \overline{x}}{s_X^2} + (x-\overline{x})^2\frac{(x_i-\overline{x})^2}{s_X^4} && (4.46)\\
&= \frac{\sigma^2}{n^2}\left(n + 0 + (x-\overline{x})^2\frac{n s_X^2}{n s_X^4}\right) && (4.47)\\
&= \frac{\sigma^2}{n}\left(1 + \frac{(x-\overline{x})^2}{s_X^2}\right) && (4.48)
\end{aligned}$$

Notice what's going on here:

- The variance grows as σ^2 grows: the more noise there is around the regression line, the harder we find it to estimate the regression line, and the more of that noise will propagate into our predictions.

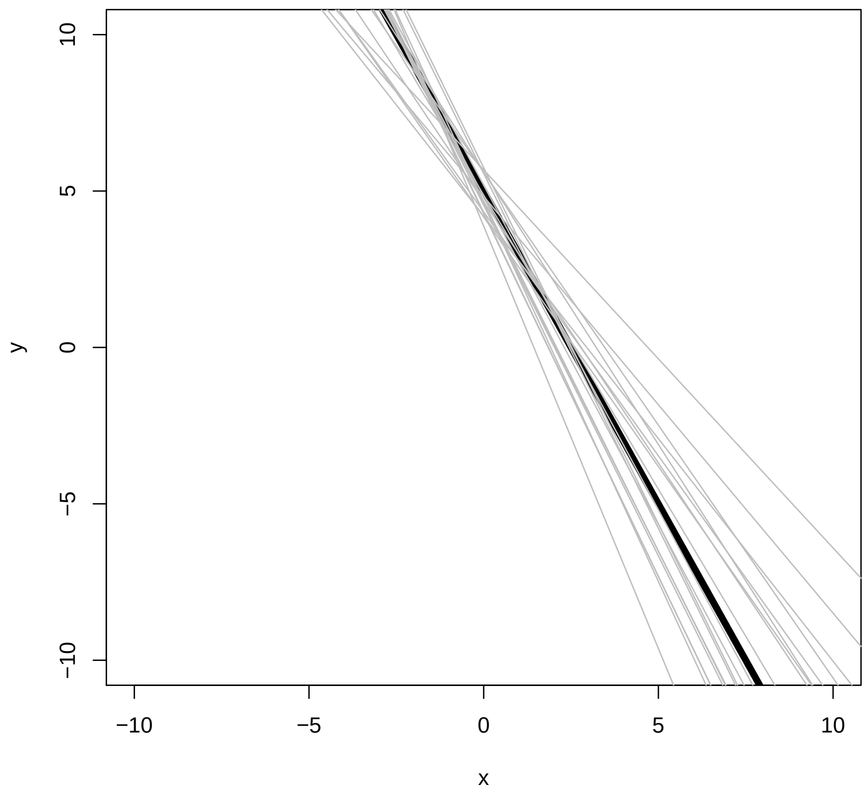
- The larger n is, the smaller the variance: the more points we see, the more exactly we can pin down the line, and so our predictions.

- The variance of our predictions is the sum of two terms. The first, which doesn't depend on x, is σ^2/n, which is the variance of $\overline{y}$ (Exercise 3). Since our line has to go through the center of the data, this just how much noise there is in the height of that center.

- The other term does change with x, specifically with $(x-\overline{x})^2$: the further our operating point x is from the center of the data $\overline{x}$, the bigger our uncertainty. This is the uncertainty that comes with being unable to pin down the slope precisely; it therefore shrinks with s_X^2, since it's easier to find the slope when the points have a wide spread on the horizontal axis.

Again, Eq. 4.48 is conditional on the x_i. If those are random, we need to use the law of total variance (as in Chapter 3) to get the unconditional variance of $\hat{m}(x)$.

Figure 4.3 illustrates how the spread in point predictions changes as we move away from the mean of the x values.



```
# Create an empty plot (type='n' for 'do Nothing')
plot(0, type = "n", xlim = c(-10, 10), ylim = c(-10, 10), xlab = "x", ylab = "y")
# Add the true regression line; exaggerate width so it stands out
abline(a = 5, b = -2, lwd = 5)
# Repeat 10 times: do a simulation, fit a line to the sim., add the fitted
# line to the plot
invisible(replicate(20, abline(lm(y ~ x, data = sim.linmod(n = 10, beta.0 = 5,
    beta.1 = -2, width = 4, df = 3)), col = "grey")))
```

FIGURE 4.3: *Scatter of estimated least-squares regression lines (thin, grey) around the true regression line (thick, black). Notice how the estimated lines become more spread out as we move away from the center of the distribution (here conveniently set at $X = 0$).*

4.5 Estimating σ^2; Sum of Squared Errors

Under the simple linear regression model, it is easy to show (Exercise 5) that

$$\mathbb{E}\left[(Y-(\beta_0+\beta_1 X))^2\right]=\sigma^2 \tag{4.49}$$

This suggests that the minimal value of the in-sample MSE,

$$\hat{\sigma}^2=\frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{m}(x_i))^2 \tag{4.50}$$

is a natural estimator for σ^2. This is, in fact, a consistent estimator. (You can prove this using the consistency of $\hat{\beta}_0$ and $\hat{\beta}_1$, and continuity.) It is, however, a slightly biased estimator. Specifically (Exercise 6)

$$s^2=\frac{n}{n-2}\hat{\sigma}^2 \tag{4.51}$$

is an *un*-biased estimator of σ^2, though one with a larger variance. Some people, accordingly, use Eq. 4.51, rather than Eq. 4.50, as their definition of "MSE".

This is mostly something to be aware of when different pieces of R code, textbooks, papers, etc., differ in what they are calling "MSE"; to get sensible results, you may need to apply conversion factors in one direction or the other. As usual, if the difference between $1/n$ and $1/(n-2)$ is large enough to make a difference to your conclusions, you should really be asking yourself whether you have enough data to be doing any statistics at all.

Sum of squared errors The sum of squared errors for a fitted regression is just what it sounds like:

$$SSE=\sum_{i=1}^{n}(y_i-\hat{m}(x_i))^2=\sum_{i=1}^{n}(y_i-(\hat{\beta}_0+\hat{\beta}_1 x_i))^2 \tag{4.52}$$

It's mostly important as a historical relic, from the days when people fit regression models by hand, or with slide rules and adding machines, and so every bit of arithmetic you could avoid was a win.

4.6 Residuals

The **residual** value at a data point is the difference between the actual value of the response y_i and the fitted value $\hat{m}(x_i)$:

$$e_i=y_i-\hat{m}(x_i)=y_i-(\hat{\beta}_0+\hat{\beta}_1 x_i) \tag{4.53}$$

This may look like re-arranging the basic equation for the linear regression model,

$$\epsilon_i=Y_i-(\beta_0+\beta_1 x_i) \tag{4.54}$$

and it *is* similar, but it's *not* the same. The right-hand side of Eq. 4.54 involves the true parameters. The right-hand side of Eq. 4.53 involves the *estimated* parameters, which are different. In particular, the estimated parameters are functions of *all* the noise variables. Therefore

The residuals are not the noise terms; $e_i \neq \epsilon_i$

There are some ways in which the residuals are *like* the noise terms. For example, the residuals are always uncorrelated with the x_i:

$$\frac{1}{n}\sum_{i=1}^{n} e_i(x_i - \overline{x}) = 0 \tag{4.55}$$

However, this fact (and others like it, which you will get to prove in the homework) are consequences of the estimating equations, and are true *whether or not* the simple linear regression model is actually true. Another consequence of the estimating equations is that

$$\frac{1}{n}\sum_{i=1}^{n} e_i = 0 \tag{4.56}$$

This is *reminiscent* of $\mathbb{E}[\epsilon] = 0$, but generally $n^{-1}\sum_{i=1}^{n}\epsilon_i \neq 0$. In fact, Eq. 4.56 implies that the residuals are actually linearly dependent on each other, while the ϵ_i are not.

Despite these differences, there is enough of a relationship between the ϵ_i and the e_i that a lot of our model-checking and diagnostics will be done in terms of the residuals. You should get used to looking at them for basically any model you estimate, or even think seriously about estimating.

4.7 Limitations of Least Squares

The results in this handout, and the last, almost exhaust the theory of statistical inference for least squares estimates in the simple linear regression model[4]. Going beyond the mean and variance of parameter estimates or predicted values is pretty much impossible, using *just* least squares and the simple linear regression model.

In particular, we can't get **sampling distributions** for anything — we can't say what probability law $\hat{\beta}_1$ follows, even as a function of the true parameters $\beta_0, \beta_1, \sigma^2$. There are just too many possibilities which are compatible with the model assumptions. Since, as you remember from your mathematical statistics course, we need sampling distributions to form confidence intervals, evaluate the properties of hypothesis tests, etc., etc., there is really not much more to say about this combination of model and method.

[4]The main exception is a result called the **Gauss-Markov theorem**: the least squares estimator has smaller variance than any other unbiased estimator which is a linear function of the y_i. This was more impressive when nobody had the computing power to use nonlinear estimators. . .

Chebyshev If we absolutely must do some probability calculations under the least-squares assumptions, the best we can usually do is to invoke Chebyshev's inequality (the extra credit problem in homework 1): for any random variable Z with expected value μ and variance σ, and any $r > 0$,

$$\mathbb{P}(|Z-\mu| \geq r) \leq \frac{\mathrm{Var}[Z]}{r^2} \tag{4.57}$$

In particular, we can say that

$$\mathbb{P}\left(|Z-\mu| \geq k\sqrt{\mathrm{Var}[Z]}\right) \leq \frac{1}{k^2} \tag{4.58}$$

These probability bounds are *very* loose, so if we do try to use them to do hypothesis tests, they have very little power (equivalently: the confidence intervals we get are huge).

Asymptotic Gaussianity The right-hand side of Eq. 4.25 shows that $\hat{\beta}_1$ is β_1 plus a weighted average of the ϵ_i. If we add to the simple linear regression model the assumption that the ϵ_i are IID draws from a fixed, *not-necessarily-Gaussian* distribution, we might then try to use the central limit theorem to show that the weighted average tends towards a Gaussian as $n \to \infty$. This can be done in some generality, but it needs more delicate probability theory than the rest of what we are doing, and if the initial distribution of the ϵ_i is, say, appreciably skewed, n might have to be truly huge before the Gaussian approximation is any good[5].

4.8 Least-Squares in R

The basic command for fitting a linear model by least squares in R is `lm`. It has a huge number of options, and can do a lot more than we will ask it to here, but for our purposes we use it as follows:

```
lm(y ~ x, data = df)
```

Here `df` is a data frame containing the data we want to fit a regression to, and the first part, the **formula**, tells `lm` that we want to regress the column of `df` called `y` on the column called `x`. By default[6], `lm` always fits its regression models by least squares.

What `lm` returns is a rather complicated object. If you just print it out, it seems to be only the intercept and the slope:

```
# Make a very small simulated data set from our running examing
toy.data <- sim.linmod(n = 10, beta.0 = 5, beta.1 = -2, width = 4, df = 3)
# Fit the simple linear regression model to it by least squares
lm(y ~ x, data = toy.data)
```

[5]To those who think everything is Gaussian once $n \geq 30$, all I can say is "Bless your heart."

[6]There are ways to tweak this, some of which we'll see later in the course.

```
##
## Call:
## lm(formula = y ~ x, data = toy.data)
##
## Coefficients:
## (Intercept)            x
##       4.582       -1.834
```

In fact, `lm` has done *lots* of calculations as part of fitting the model, and stored many of the results into the object it returns; R just doesn't print all of that, unless you make it. We can see some of what's in there, though:

```
names(lm(y ~ x, data = toy.data))
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
```

(See `help(lm)`, under "Value", for the gory details.) It's annoying (and slow and error-prone) to keep having R re-estimate the model every time we want to refer back to it, so we usually store the estimated model under a new variable name:

```
# Fit a linear model to the toy data, and store as toy.lm The name of the
# estimated model needn't match that of the data, but it's often a good idea
toy.lm <- lm(y ~ x, data = toy.data)
```

We can access some of what's in the `lm` object by using special functions. A couple in particular will become close and familiar friends. `coefficients` gives us the vector of coefficients:

```
coefficients(toy.lm)
## (Intercept)           x
##    4.581803   -1.834000
```

`fitted` gives us the vector of fitted values, in the order which the data points appeared:

```
fitted(toy.lm)
##        1        2        3        4        5        6        7        8
## 6.825715 3.204983 2.733102 6.569347 8.066388 1.832514 3.466832 5.085674
##        9       10
## 3.647524 3.901256
```

`residuals` gives us the vector of residuals (ditto order):

```
residuals(toy.lm)
##           1           2           3           4           5           6
## -0.04705055  1.63707370  1.29248794 -2.23857540  1.87083552 -1.07608760
##           7           8           9          10
##  1.56471553 -1.25909275 -3.59676400  1.85245760
```

(How would you use `residuals` to calculate s^2? To calculate $\frac{n}{n-2}s^2$?)

You might think that `plot(toy.lm)` would draw a picture of the fitted model; instead, it goes through a bunch of diagnostic plots, which we will get to later. If we want to add a line based on the model to an existing plot, we use `abline`, as in Figure 4.4.

`fitted` gives us the model's predictions at the particular x_i where we happened to have data. In principle, though, the model has an opinion about what $\mathbb{E}[Y|X=x]$ should be at every possible value of x. To extract that, we use a function called `predict`. Naturally enough, we need to tell it both which model we want to use (since we could have more than one floating around), *and* where to make the predictions:

```
predict(object, newdata)
```

Here the first argument, what `predict` somewhat obscurely calls `object`, is the estimated regression model, like our `toy.lm`. (It is *not* the name of the estimating function, like `lm`.) The second argument, `newdata`, is a data frame with a column whose name matches the column name of the predictor variable in our original data frame. Thus

```
predict(toy.lm, newdata = data.frame(x = 1:5))
##          1          2          3          4          5
##  2.7478031  0.9138027 -0.9201976 -2.7541980 -4.5881983
```

gives us the fitted model's predictions at the integers from 1 to 5.

You might well think that if `newdata` were missing, then `predict` would throw an error. You might very well think that.

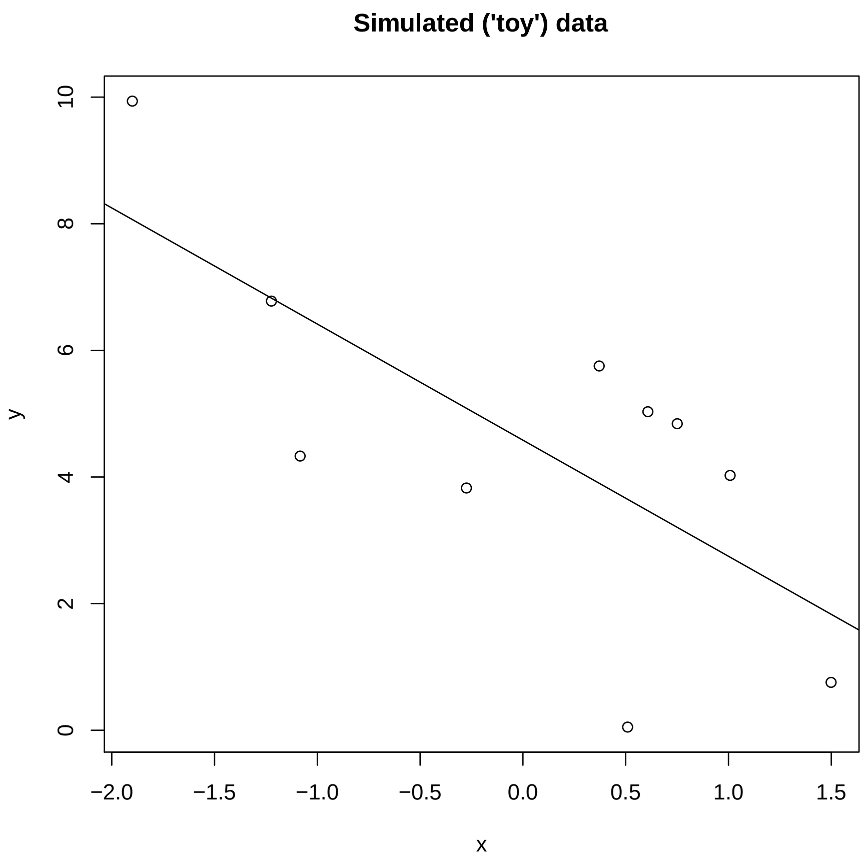
```
predict(toy.lm)
##        1        2        3        4        5        6        7        8
## 6.825715 3.204983 2.733102 6.569347 8.066388 1.832514 3.466832 5.085674
##        9       10
## 3.647524 3.901256
predict(toy.lm, data = data.frame(x = 1:5))  # What's wrong here?
##        1        2        3        4        5        6        7        8
## 6.825715 3.204983 2.733102 6.569347 8.066388 1.832514 3.466832 5.085674
##        9       10
## 3.647524 3.901256
```

For reasons best known to the designers of R[7], when `newdata` is missing or malformed, `predict` returns the fitted values on the original data. On the other hand, you *will* get an error if `newdata` exists but doesn't contain the right column name:

```
predict(toy.lm, newdata = data.frame(zebra = 1:5))

## Error in eval(predvars, data, env):  object 'x' not found
```

Extraneous columns, however, are just ignored:

[7] Really, the designers of the predecessor language, S.



```
plot(y ~ x, data = toy.data, xlab = "x", ylab = "y", main = "Simulated ('toy') data")
abline(toy.lm)
```

FIGURE 4.4: *Using* `abline` *to add the line of an estimated linear regression model to a plot.*

```
predict(toy.lm, newdata = data.frame(x = 1:5, zebra = 6:10))
##          1          2          3          4          5
##  2.7478031  0.9138027 -0.9201976 -2.7541980 -4.5881983
```

There is one further option to `predict` which is worth mentioning at this time. If we set `se.fit=TRUE`, we get the standard errors of the fitted values, i.e., the square roots of the variances[8]:

```
predict(toy.lm, newdata = data.frame(x = 1:5), se.fit = TRUE)
## $fit
##          1          2          3          4          5
##  2.7478031  0.9138027 -0.9201976 -2.7541980 -4.5881983
##
## $se.fit
##         1         2         3         4         5
## 0.8960963 1.4006427 1.9766153 2.5765505 3.1869442
##
## $df
## [1] 8
##
## $residual.scale
## [1] 2.074365
```

Notice that what this gives us back is not a vector but a *list*, whose first two entries are vectors. (We will come back to what the `df` means, but you should already be able to guess what `residual.scale` might be.)

4.9 Propagation of Error, *alias* "The Delta Method"

An optional section, but a very useful one.

The constant-plus-sum-of-noise-terms trick is the core of an extremely handy technique for getting approximate variances and standard errors for functions of quantities which are themselves estimated with error. It is known variously as "propagation of error" or (more obscurely) as "the delta method".

Suppose we are trying to estimate some quantity θ. We compute an estimate $\widehat{\theta}$, based on our data. Since our data is more or less random, so is $\widehat{\theta}$. One convenient way of measuring the purely statistical noise or uncertainty in $\widehat{\theta}$ is its standard deviation. This is the **standard error** of our estimate of θ.[9] Standard errors are not the only way of summarizing this noise, nor a completely sufficient way, but they are often useful.

[8]If a homework problem asks you to calculate the variance of a predicted value, it's (generally) asking you to do the character-building work of actually putting numbers into an algebraic formula by yourself, though you can use this to check your work.

[9]It is not, of course, to be confused with the standard deviation of the data. It is not even to be confused with the standard error of the mean, unless θ is the expected value of the data and $\widehat{\theta}$ is the sample mean.

Suppose that our estimate $\widehat{\theta}$ is a function of some intermediate quantities $\widehat{\psi_1}, \widehat{\psi_2}, \ldots, \widehat{\psi_p}$, which are also estimated:

$$\widehat{\theta} = f(\widehat{\psi_1}, \widehat{\psi_2}, \ldots \widehat{\psi_p}) \tag{4.59}$$

For instance, θ might be the difference in expected values between two groups, with ψ_1 and ψ_2 the expected values in the two groups, and $f(\psi_1, \psi_2) = \psi_1 - \psi_2$. If we have a standard error for each of the original quantities $\widehat{\psi_i}$, it would seem like we should be able to get a standard error for the **derived quantity** $\widehat{\theta}$. Propagation of error achieves this, by writing $\widehat{\theta}$ in the constant-plus-noise form.

We start with a Taylor expansion. We'll write ψ_i^* for the true (population, distribution, ensemble) value which is estimated by $\widehat{\psi_i}$.

$$\begin{aligned}
f(\psi_1^*, \psi_2^*, \ldots \psi_p^*) &\approx f(\widehat{\psi_1}, \widehat{\psi_2}, \ldots \widehat{\psi_p}) + \sum_{i=1}^{p} (\psi_i^* - \widehat{\psi_i}) \left.\frac{\partial f}{\partial \psi_i}\right|_{\psi=\widehat{\psi}} && (4.60)\\
f(\widehat{\psi_1}, \widehat{\psi_2}, \ldots \widehat{\psi_p}) &\approx f(\psi_1^*, \psi_2^*, \ldots \psi_p^*) + \sum_{i=1}^{p} (\widehat{\psi_i} - \psi_i^*) \left.\frac{\partial f}{\partial \psi_i}\right|_{\psi=\widehat{\psi}} && (4.61)\\
\hat{\theta} &\approx \theta^* + \sum_{i=1}^{p} (\widehat{\psi_i} - \psi_i^*) f_i'(\widehat{\psi}) && (4.62)
\end{aligned}$$

introducing f_i' as an abbreviation for $\frac{\partial f}{\partial \psi_i}$. The left-hand side is now the quantity whose standard error we want. I have done this manipulation because now $\hat{\theta}$ is a linear function (approximately!) of some random quantities whose variances we know, and some derivatives which we can calculate.

Remember (from Chapter 1) the rules for arithmetic with variances: if X and Y are random variables, and a, b and c are constants,

$$\begin{aligned}
\mathrm{Var}[a] &= 0 && (4.63)\\
\mathrm{Var}[a + bX] &= b^2 \mathrm{Var}[X] && (4.64)\\
\mathrm{Var}[a + bX + cY] &= b^2 \mathrm{Var}[X] + c^2 \mathrm{Var}[Y] + 2bc\mathrm{Cov}[X, Y] && (4.65)
\end{aligned}$$

While we don't know $f(\psi_1^*, \psi_2^*, \ldots \psi_p^*)$, it's constant, so it has variance 0. Similarly, $\mathrm{Var}\left[\widehat{\psi_i} - \psi_i^*\right] = \mathrm{Var}\left[\widehat{\psi_i}\right]$. Repeatedly applying these rules to Eq. 4.62,

$$\mathrm{Var}\left[\widehat{\theta}\right] \approx \sum_{i=1}^{p} (f_i'(\widehat{\psi}))^2 \mathrm{Var}\left[\widehat{\psi_i}\right] + 2\sum_{i=1}^{p-1} \sum_{j=i+1}^{p} f_i'(\widehat{\psi}) f_j'(\widehat{\psi}) \mathrm{Cov}\left[\widehat{\psi_i}, \widehat{\psi_j}\right] \tag{4.66}$$

The standard error for $\widehat{\theta}$ would then be the square root of this.

If we follow this rule for the simple case of group differences, $f(\psi_1, \psi_2) = \psi_1 - \psi_2$, we find that

$$\mathrm{Var}\left[\widehat{\theta}\right] = \mathrm{Var}\left[\widehat{\psi_1}\right] + \mathrm{Var}\left[\widehat{\psi_2}\right] - 2\mathrm{Cov}\left[\widehat{\psi_1}, \widehat{\psi_2}\right] \tag{4.67}$$

just as we would find from the basic rules for arithmetic with variances. The approximation in Eq. 4.66 comes from the nonlinearities in f.

If the estimates of the initial quantities are uncorrelated, Eq. 4.66 simplifies to

$$\text{Var}\left[\widehat{\theta}\right] \approx \sum_{i=1}^{p} (f_i'(\widehat{\psi}))^2 \text{Var}\left[\widehat{\psi_i}\right] \tag{4.68}$$

and, again, the standard error of $\widehat{\theta}$ would be the square root of this. The special case of Eq. 4.68 is sometimes called *the* propagation of error formula, but I think it's better to use that name for the more general Eq. 4.66.

Exercises

1. *True MSE of a linear model* Prove that the full-distribution MSE of a linear model with intercept b_0 and slope b_1 is

$$\text{Var}[Y] + (\mathbb{E}[Y])^2 - 2b_0\mathbb{E}[Y] - 2b_1\text{Cov}[X,Y] - 2b_1\mathbb{E}[X]\mathbb{E}[Y] + 2b_1\mathbb{E}[X] + b_1^2\text{Var}[X] + b_1^2(\mathbb{E}[X])^2 \tag{4.69}$$

2. Show that if all $x_i = \overline{x}$, then the system of linear equations, Eqs. 4.2–4.3, actually contains only one linearly-independent equation. *Hint:* Write the system of equations as a matrix multiplying the vector whose entries are $(\hat{\beta}_0, \hat{\beta}_1)$, and consider the determinant of the matrix. (How does the determinant of such a matrix relate to whether the equations are all linearly independent?)

3. Show that, under the simple linear regression model, $\text{Var}[\overline{y}] = \sigma^2/n$. You may treat the x_i as fixed in this calculation.

4. Derive Eqs. 4.6 and 4.9 from the results in §4.4. (Hint: $\hat{\beta}_0 = \hat{m}(x)$ for what value of x?) Is this a circular argument?

5. Prove Eq. 4.49.

6. Express the right-hand side of Eq. 4.50 in terms of β_0, β_1, the x_i and the ϵ_i. Use this expression to find $\mathbb{E}\left[\hat{\sigma}^2\right]$, and show that it equals $\frac{n-2}{n}\sigma^2$.

Chapter 5

The Method of Maximum Likelihood for Simple Linear Regression

5.1 Recapitulation

We introduced the method of maximum likelihood for simple linear regression in Chapter 3. Let's review.

We start with the statistical model, which is the Gaussian-noise simple linear regression model, defined as follows:

1. The distribution of X is arbitrary (and perhaps X is even non-random).
2. If $X = x$, then $Y = \beta_0 + \beta_1 x + \epsilon$, for some constants ("coefficients", "parameters") β_0 and β_1, and some random noise variable ϵ.
3. $\epsilon \sim N(0, \sigma^2)$, and is independent of X.
4. ϵ is independent across observations.

A consequence of these assumptions is that the response variable Y is independent across observations, conditional on the predictor X, i.e., Y_1 and Y_2 are independent given X_1 and X_2 (Exercise 1).

As you'll recall, this is a special case of the simple linear regression model: the first two assumptions are the same, but we are now assuming much more about the noise variable ϵ: it's not just mean zero with constant variance, but it has a particular distribution (Gaussian), and everything we said was uncorrelated before we now strengthen to independence[1].

[1]See Chapter 1 for a reminder, with an explicit example, of how uncorrelated random variables can nonetheless be strongly statistically dependent.

Because of these stronger assumptions, the model tells us the conditional pdf of Y for each x, $p(y|X=x;\beta_0,\beta_1,\sigma^2)$. (This notation separates the random variables from the parameters.) Given any data set $(x_1,y_1),(x_2,y_2),\ldots(x_n,y_n)$, we can now write down the probability density, under the model, of seeing that data:

$$\prod_{i=1}^{n} p(y_i|x_i;\beta_0,\beta_1,\sigma^2) = \prod_{i=1}^{n}\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i-(\beta_0+\beta_1 x_i))^2}{2\sigma^2}}$$

In multiplying together the probabilities like this, we are using the conditional independence of the Y_i (given the X_i), which follows from the independence of the ϵ_i.

When we see the data, we do not *known* the true parameters, but any guess at them, say (b_0,b_1,s^2), gives us a probability density:

$$\prod_{i=1}^{n} p(y_i|x_i;b_0,b_1,s^2) = \prod_{i=1}^{n}\frac{1}{\sqrt{2\pi s^2}} e^{-\frac{(y_i-(b_0+b_1 x_i))^2}{2s^2}}$$

This is the **likelihood**, a function of the parameter values. It's just as informative, and much more convenient, to work with the **log-likelihood**,

$$L(b_0,b_1,s^2) = \log\prod_{i=1}^{n} p(y_i|x_i;b_0,b_1,s^2) \quad (5.1)$$

$$= \sum_{i=1}^{n}\log p(y_i|x_i;b_0,b_1,s^2) \quad (5.2)$$

$$= -\frac{n}{2}\log 2\pi - n\log s - \frac{1}{2s^2}\sum_{i=1}^{n}(y_i-(b_0+b_1x_i))^2 \quad (5.3)$$

In the **method of maximum likelihood**, we pick the parameter values which maximize the likelihood, or, equivalently, maximize the log-likelihood. After some calculus (see Chapter 3, this gives us the following estimators:

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i-\overline{x})(y_i-\overline{y})}{\sum_{i=1}^{n}(x_i-\overline{x})^2} = \frac{c_{XY}}{s_X^2} \quad (5.4)$$

$$\hat{\beta}_0 = \overline{y} - \hat{beta}_1\overline{x} \quad (5.5)$$

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i-(\hat{\beta}_0+\hat{\beta}_1x_i))^2 \quad (5.6)$$

As you will recall, the estimators for the slope and the intercept exactly match the least squares estimators. This is a special property of assuming independent Gaussian noise. Similarly, $\hat{\sigma}^2$ is exactly the in-sample mean squared error.

5.2 Sampling Distributions

We may seem not to have gained much from the Gaussian-noise assumption, because our point estimates are just the same as they were from least squares. What makes the

Gaussian noise assumption important is that it gives us an exact conditional distribution for each Y_i, and this in turn gives us a distribution — the **sampling distribution** — for the estimators. Remember, from the notes from last time, that we can write $\hat{\beta}_1$ and $\hat{\beta}_0$ in the form "constant plus sum of noise variables". For instance,

$$\hat{\beta}_1 = \beta_1 + \sum_{i=1}^{n} \frac{x_i - \overline{x}}{n s_X^2} \epsilon_i$$

Now, in the Gaussian-noise model, the ϵ_i are all independent Gaussians. Therefore, $\hat{\beta}_1$ *is also Gaussian*. Since we worked out its mean and variance last time, we can just say

$$\hat{\beta}_1 \sim N(\beta_1, \sigma^2 / n s_X^2)$$

Again, we saw that the fitted value at an arbitrary point x, $\hat{m}(x)$, is a constant plus a weighted sum of the ϵ:

$$\hat{m}(x) = \beta_0 + \beta_1 x + \frac{1}{n} \sum_{i=1}^{n} \left(1 + (x - \overline{x}) \frac{x_i - \overline{x}}{s_X^2}\right) \epsilon_i$$

Once again, because the ϵ_i are independent Gaussians, a weighted sum of them is also Gaussian, and we can just say

$$\hat{m}(x) \sim N\left(\beta_0 + \beta_1 x, \frac{\sigma^2}{n}\left(1 + \frac{(x - \overline{x})^2}{s_X^2}\right)\right)$$

Slightly more complicated manipulation of the ϵ_i makes it possible to show that

$$\frac{n \hat{\sigma}^2}{\sigma^2} \sim \chi^2_{n-2}$$

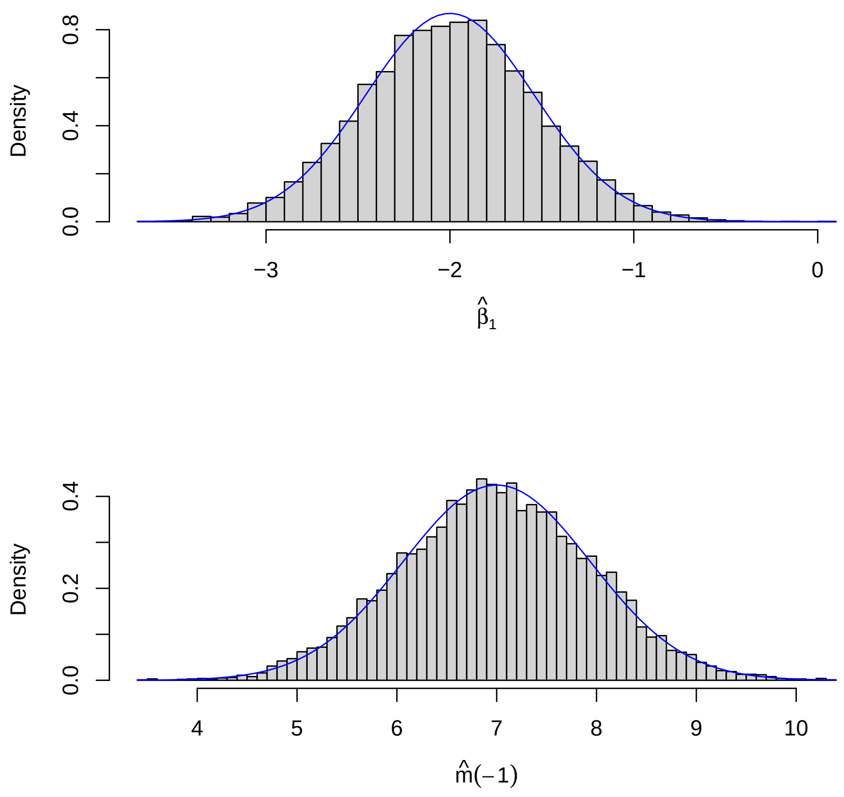
These are all important, because when we come to doing statistical inference on the parameters — forming confidence intervals, or testing hypotheses — we need to know these sampling distributions. When we come to making predictions of new Y's, these sampling distributions will let us give confidence intervals for the expected values, $\hat{m}(x)$, as well as give prediction intervals (of the form "when $X = 5$, Y will be between l and u with 95% probability") or full distributional forecasts. We will derive these inferential formulas in later chapters.

5.2.1 Illustration

To make the idea of these sampling distributions more concrete, I present a small simulation. Figure 5.1 provides code which simulates a particular Gaussian-noise linear model: $\beta_0 = 5$, $\beta_1 = -2$, $\sigma^2 = 3$, with twenty X's initially randomly drawn from an exponential distribution, but thereafter held fixed through all the simulations. The theory above lets us calculate just what the distribution of $\hat{\beta}_1$ should be, in repeated simulations, and the distribution of $\hat{m}(-1)$. (By construction, we have no *observations* where $x = -1$; this is an example of using the model to extrapolate beyond the data.) Figure 5.2 compares the theoretical sampling distributions to what we actually get by repeated simulation, i.e., by repeating the experiment.

```
# Fix x values for all runs of the simulation; draw from an exponential
n <- 20  # So we don't have magic #s floating around
beta.0 <- 5
beta.1 <- -2
sigma.sq <- 3
fixed.x <- rexp(n = n)

# Simulate from the model Y=\beta_0+\beta_1*x+N(0,\sigma^2) Inputs: intercept;
# slope; variance; vector of x; return sample or estimated linear model?
# Outputs: data frame with columns x and y OR linear model fit to simulated y
# regressed on x
sim.lin.gauss <- function(intercept = beta.0, slope = beta.1, noise.variance = sigma.sq,
    x = fixed.x, model = FALSE) {
    # Make up y by adding Gaussian noise to the linear function
    y <- rnorm(length(x), intercept + slope * x, sd = sqrt(noise.variance))
    # Do we want to fit a model to this simulation and return that model?  Or
    # do we want to just return the simulated values?
    if (model) {
        return(lm(y ~ x))
    } else {
        return(data.frame(x = x, y = y))
    }
}
```

FIGURE 5.1: *Function to simulate a Gaussian-noise simple linear regression model, together with some default parameter values. Since, in this chapter, we'll always be estimating a linear model on the simulated values, it makes sense to build that into the simulator, but I included a switch to control that.*



```
par(mfrow = c(2, 1))
slope.sample <- replicate(10000, coefficients(sim.lin.gauss(model = TRUE))["x"])
hist(slope.sample, freq = FALSE, breaks = 50, xlab = expression(hat(beta)[1]), main = "")
curve(dnorm(x, -2, sd = sqrt(3/(n * var(fixed.x)))), add = TRUE, col = "blue")
pred.sample <- replicate(10000, predict(sim.lin.gauss(model = TRUE), newdata = data.frame(x = -1)))
hist(pred.sample, freq = FALSE, breaks = 50, xlab = expression(hat(m)(-1)), main = "")
curve(dnorm(x, mean = beta.0 + beta.1 * (-1), sd = sqrt((sigma.sq/n) * (1 + (-1 -
    mean(fixed.x))^2/var(fixed.x)))), add = TRUE, col = "blue")
```

FIGURE 5.2: *Theoretical sampling distributions for $\hat{\beta}_1$ and $\hat{m}(-1)$ (blue curves) versus the distribution in 10^4 simulations (black histograms).*

5.3 Virtues and Limitations of Maximum Likelihood

The method of maximum likelihood does not always work; there are models where it gives poor or even pathological estimates. For Gaussian-noise linear models, however, it actually works very well. Indeed, in more advanced statistics classes, one proves that for such models, as for many other "regular" statistical models, maximum likelihood is **asymptotically efficient**, meaning that its parameter estimates converge on the truth as quickly as possible[2]. This is on top of having exact sampling distributions for the estimators.

Of course, all these wonderful abilities come at a cost, which is the Gaussian noise assumption. If that is wrong, then so are the sampling distributions I gave above, and so are the inferential calculations which rely on those sampling distributions. *Before* we begin to do those inferences on any particular data set, and *especially* before we begin to make grand claims about the world on the basis of those inferences, we should really check all those modeling assumptions. That, however, brings us into the topics for next week.

Exercises

1. Let $Y_1, Y_2, \ldots Y_n$ be generated from the Gaussian-noise simple linear regression model, with the corresponding values of the predictor variable being $X_1, \ldots X_n$. Show that if $i \neq j$, then Y_i and Y_j are conditionally independent given (X_i, X_j). *Hint:* If U and V are independent, then $f(U)$ and $g(V)$ are also independent, for any functions f and g.

2. In many practical fields (e.g., finance and geology) it is common to encounter noise whose distribution has much heavier tails than any Gaussian could give us. One way to model this is with t distributions. Consider, therefore, the statistical model where $Y = \beta_0 + \beta_1 X + \epsilon$, and $\epsilon/\sigma \sim t_\nu$, with ϵ independent of X and independent across observations. That is, rather than having a Gaussian distribution, the noise follows a t distribution with ν degrees of freedom (after scaling).

   Note: Most students find most parts after (a) quite challenging.

   (a) Write down the log-likelihood function. Use an explicit formula for the density of the t distribution.

   (b) Find the derivatives of this log-likelihood with respect to the four parameters β_0, β_1, σ (or σ^2, if more convenient) and ν. Simplify as much as possible. (It is legitimate to use derivatives of the gamma function here, since that's another special function.)

[2] *Very* roughly: writing θ for the true parameter, $\hat{\theta}$ for the MLE, and $\tilde{\theta}$ for any other consistent estimator, asymptotic efficiency means $\lim_{n\to\infty} \mathbb{E}\left[n\|\hat{\theta}-\theta\|^2\right] \leq \lim_{n\to\infty} \mathbb{E}\left[n\|\tilde{\theta}-\theta\|\right]$. (This way of formulating it takes it for granted that the MSE of estimation goes to zero like $1/n$, but it typically does in parametric problems.) For more precise statements, see, for instance, Cramér (1945), Pitman (1979) or van der Vaart (1998).

(c) Can you solve for the maximum likelihood estimators of β_0 and β_1 without knowing σ and ν? If not, why not? If you can, do they match the least-squares estimators again? If they don't match, how do they differ?

(d) Can you solve for the MLE of all four parameters at once? (Again, you may have to express your answer in terms of the gamma function and its derivatives.)

3. Refer to the previous problem, and do part (a).

   (a) In R, write a function to calculate the log-likelihood, taking as arguments a data frame with columns names `y` and `x`, and the vector of the four model parameters. *Hint:* use the `dt` function.

   (b) In R, using `optim`, write a function to find the MLE of this model on a given data set, from an arbitrary starting vector of guesses at the parameters. This should call your function from part (a).

   (c) In R, write a function which gives an unprincipled but straight-forward initial estimate of the parameters by (i) calculating the slope and intercept using least squares, and (ii) fitting a t distribution to the residuals. *Hint:* call `lm`, and `fitdistr` from the package `MASS`.

   (d) Combine your functions to write a function which takes as its only argument a data frame containing columns called `x` and `y`, and returns the MLE for the model parameters.

   (e) Write another function which will simulte data from the model, taking as arguments the four parameters and a vector of x's. It should return a data frame with appropriate column names.

   (f) Run the output of your simulation function through your MLE function. How well does the MLE recover the parameters? Does it get better as n grows? As the variance of your x's increases? How does it compare to your unprincipled estimator?

Chapter 6

Diagnostics and Modifications for Simple Regression

In the previous chapters, we have laid out what regression analysis is for, why we use statistical models to do it, the assumptions of the simple linear regression model, and estimation and prediction for the simple regression model using both the method of maximum likelihood and the method of least squares. We could, at this point, follow the traditional route in a class like this of plunging into detailed inferential calculations for the model (this is how you find a confidence interval for such-and-such, etc.). However, those calculations have little point if the assumptions for the model do not hold. Thus, we will look at model checking *first*, and in later chapters go on to the inferential theory.

It is somewhat more traditional to talk about "diagnostics" for the regression model, rather than "model checking"; which name is best depends on how formal you want to sound. Just as in medicine, a good diagnostic procedure is one which will *differentiate* between health (for us: the model assumptions hold) and illness (the assumptions are violated). Indeed, a really good diagnostic procedure will distinguish between different sorts of illness (tell us *which* assumptions are violated). More mathematically, we want to find consequences of the assumptions which will hold no matter what the parameter values might be, but would be hard to achieve if the assumptions fail. (Checking whether $\sum_i \hat{\beta}_0 + \hat{\beta}_1 x_i = \sum_i y_i$ is not a useful diagnostic, because while that will be true if the assumptions are all right, it would still be true if they were all wrong[1].) The simplest and most useful check all involve the residuals.

6.1 The Residuals

In previous chapters, we defined the **residual** at the i^{th} data point as the difference between the realized value of the response y_i and what the estimated model would

[1]I could give you a demonstration of that equation, but that would spoil one of the problems in the current homework.

predict:

$$e_i = y_i - \hat{m}(x_i) \tag{6.1}$$

which, for the simple linear regression model, is just

$$e_i = y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i) \tag{6.2}$$

Residuals vs. the noise The residuals are very closely related to the noise variables ϵ_i, but with some important differences. If we take the basic equation for the simple linear model, $Y_i = \beta_0 + \beta_1 x_i + \epsilon_i$, we can re-arrange it to read

$$\epsilon_i = Y_i - (\beta_0 + \beta_1 x_i) \tag{6.3}$$

This has the same form as the equation for the residuals, but it involves the true parameters, not their estimates. If we take that equation for the i^{th} residual and substitute in the equation for Y_i,

$$\begin{aligned} e_i &= \beta_0 + \beta_1 x_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i) + \epsilon_i & (6.4)\\ &= (\beta_0 - \hat{\beta}_0) + (\beta_1 - \hat{\beta}_1) x_i + \epsilon_i & (6.5) \end{aligned}$$

The terms in parentheses on the right-hand side are hopefully small, but they're not (in general) zero.

Residuals as weighted sums of noise To understand what's going on with the residuals, it's helpful to write them as a weighted sum of the ϵ's. Going back to previous chapters,

$$\begin{aligned} \hat{\beta}_0 &= \beta_0 + \frac{1}{n}\sum_{i=1}^{n}\left(1 - \overline{x}\frac{x_i - \overline{x}}{s_X^2}\right)\epsilon_i & (6.6)\\ \hat{\beta}_1 &= \beta_1 + \sum_{i=1}^{n}\frac{x_i - \overline{x}}{n s_X^2}\epsilon_i & (6.7) \end{aligned}$$

Substitute these in to the equation for e_i:

$$\begin{aligned} e_i &= \epsilon_i + \frac{1}{n}\sum_{j=1}^{n}\left(1 - \overline{x}\frac{x_j - \overline{x}}{s_X^2}\right)\epsilon_j + x_i \sum_{j=1}^{n}\frac{x_j - \overline{x}}{n s_X^2}\epsilon_j & (6.8)\\ &= \sum_{j=1}^{n}\left(\delta_{ij} + \frac{1}{n} + (x_i - \overline{x})\frac{x_j - \overline{x}}{n s_X^2}\right)\epsilon_j & (6.9) \end{aligned}$$

using the "Kronecker delta" ($\delta_{ij} = 0$ if $i \neq j$, $= 1$ if $i = j$) and some algebra. The factor in parenthesis is a weight which gets applied to each ϵ_j when summing them up — a weight which depends on the x's alone. Let's abbreviate this weight by c_{ij}.

One of the assumptions of the simple linear model is that $\mathbb{E}[\epsilon_i|X] = 0$. Since we've shown that e_i is a weighted sum of ϵ_j, it follows that

$$\mathbb{E}[e_i|X] = \sum_j c_{ij}\mathbb{E}\left[\epsilon_j|X\right] = 0 \tag{6.10}$$

Since the simple linear model assumes that the ϵ's are uncorrelated and all have variance σ^2, even conditional on X,

$$\begin{aligned} \text{Var}[e_i|X] &= \sum_j \text{Var}\left[c_{ij}\epsilon_j|X\right] &(6.11)\\ &= \sum_j c_{ij}^2 \text{Var}\left[\epsilon_j|X\right] &(6.12)\\ &= \sigma^2 \sum_{j=1}^{n} c_{ij}^2 &(6.13)\end{aligned}$$

(I will not bother writing out the sum explicitly.) From here, one can go on to show

$$\text{Var}[e_i] = \frac{n-2}{n}\sigma^2 \tag{6.14}$$

though again I omit the details, so as not to spoil a future assignment.

If we make the Gaussian noise assumption, the ϵ_j are independent Gaussians. It thus follows that e_i also has a Gaussian distribution.

Contrast between residuals and noise terms The sum of the noise terms which produced the data is rarely zero. The *expectation value* of the sum of the noise is zero,

$$\mathbb{E}\left[\sum_{i=1}^{n} \epsilon_i\right] = \sum_{i=1}^{n} \mathbb{E}[\epsilon_i] = 0 \tag{6.15}$$

but the *variance* is not:

$$\text{Var}\left[\sum_{i=1}^{n} \epsilon_i\right] = \sum_{i=1}^{n} \text{Var}[\epsilon_i] = n\sigma^2 \tag{6.16}$$

so the sum of the noise terms can't be exactly zero all the time:

$$\sum_{i=1}^{n} \epsilon_i \neq 0 \tag{6.17}$$

(Indeed, if the ϵ_i follow a Gaussian distribution, then $\sum_{i=1}^{n} \epsilon_i \sim N(0, n\sigma^2)$, and the probability that $\sum_{i=1}^{n} \epsilon_i = 0$ is zero, not one.) Similarly, while the ϵ are uncorrelated with X,

$$\text{Cov}[X,\epsilon] = \mathbb{E}[X\epsilon] - \mathbb{E}[\epsilon]\mathbb{E}[X] = 0 \tag{6.18}$$

the ϵ_i don't have a zero *sample* correlation with X:

$$\sum_{i=1}^{n} \epsilon_i(x_i - \overline{x}) \neq 0 \tag{6.19}$$

On the other hand, such equations do hold exactly and deterministically for the residuals. In particular,

$$\sum_{i=1}^{n} e_i = 0 \tag{6.20}$$

when the simple linear model is estimated by least squares, no matter what. This is because this equation is a consequence of the estimating equations and the estimating equations alone — it applies all the time, on any data set, not just with probability 1 but without exception. Similarly,

$$\sum_{i=1}^{n}(x_i - \overline{x})e_i = 0 \tag{6.21}$$

also follows from the estimating equations. I will not go over the derivations, so as not to spoil the homework you are currently doing.

(If we think of $(e_1, e_2, \ldots e_n)$ as an n-dimensional vector, these two equations tell us that not every vector is possible. Only vectors which live in an $(n-2)$-dimensional subspace are allowed. This is, so to speak, the same $n-2$ as in $\mathrm{Var}[e_i]$.)

These two equations imply that even when the ϵ's are independent, the residuals are not. (After all, if we know all but one residual, the last one is completely determined.) However, the dependence is typically very slight and subtle, and it gets weaker as n grows (because each e_i is making a comparatively-small contribution to the sum that must be zero). So the residuals should show only negligible correlation (Exercise 1).

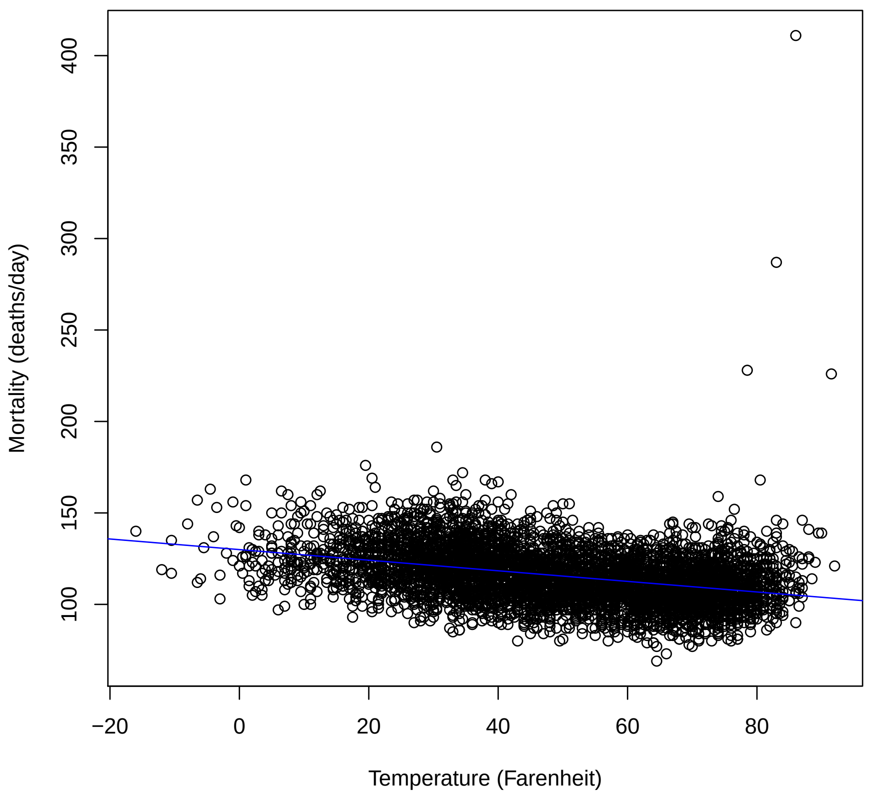
6.1.1 Summary on Properties of the Residuals

Let's sum up the most relevant observations from the last couple of paragraphs.

1. The residuals should have expectation zero, conditional on x, $\mathbb{E}[e_i|X = x] = 0$. (The residuals should also have an over-all sample mean of exactly zero.)
2. The residuals should show a nearly-constant variance[2].
3. The residuals can't be completely uncorrelated with each other, but the correlation should be extremely weak, and grow negligible as $n \to \infty$.
4. If the noise is Gaussian, the residuals should also be Gaussian.

Each one of these points leads to a diagnostic, to a check on the model. These take the form of our plots, which you should always, always make for any regression you run.

[2]We've actyally seen a formula which lets us work out the variance of the residuals — which one?



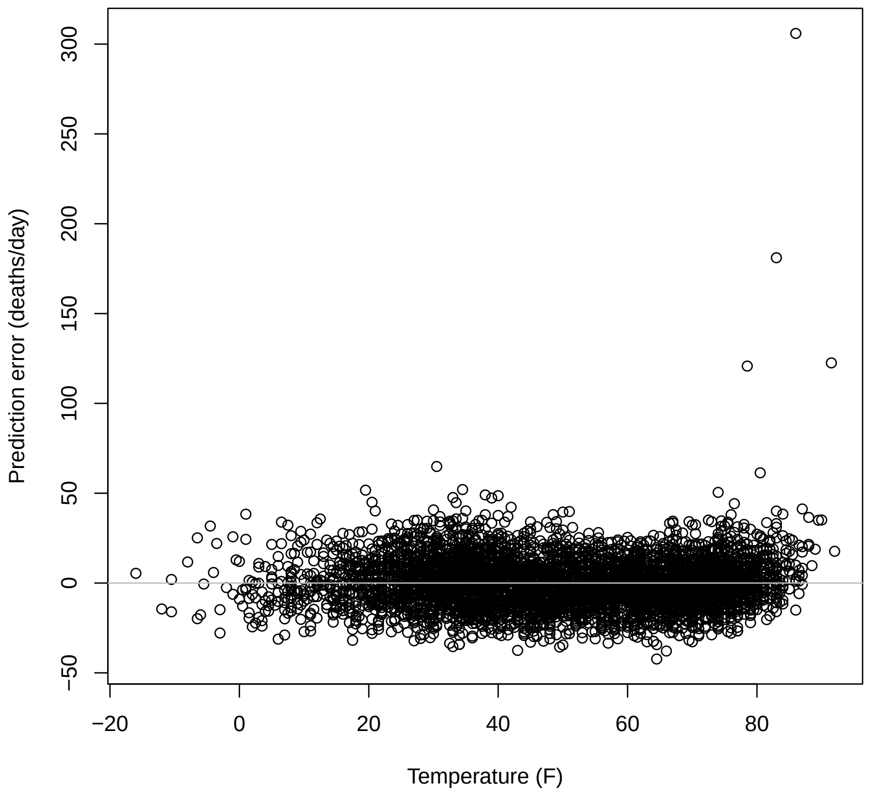
```
# Load the data set from the last homework
library(gamair)
data(chicago)

# Plot deaths each day vs. temperature
plot(death ~ tmpd, data = chicago, xlab = "Temperature (Farenheit)", ylab = "Mortality (deaths/day)")

# Estimate and store a linear model
death.temp.lm <- lm(death ~ tmpd, data = chicago)
abline(death.temp.lm, col = "blue")
```

FIGURE 6.1: *Plot of the data from the last homework, used as a running example, along with the estimated linear model (in blue).*

6.1.2 Plot the Residuals Against the Predictor

Make a scatter-plot with the residuals on the vertical axis and the predictor variable on the horizontal axis. Because $\mathbb{E}[e|X = x] = 0$, and $\mathrm{Var}[e|X = x]$ is constant, this should, ideally, look like a constant-width blur of points around a straight, flat line at height zero. Deviations from this — changing width, curvature, substantial regions of the x axis where the average residuals are either positive or negative — are all signs that the model is mis-specified. In particular, curved or stepped patterns indicate that $\mathbb{E}[e|X = x] \neq 0$, which in turn means that $\mathbb{E}[\epsilon|X = x] \neq 0$, which means that the simple-linear part of the simple linear regression model is wrong[3]. One needs either more predictor variables (getting us into multiple regression), or a different functional form for the regression (getting us into nonlinear regression), or both.

[3]See §6.5.



```
# Always plot residuals vs. predictor variable
plot(chicago$tmpd, residuals(death.temp.lm), xlab = "Temperature (F)", ylab = "Prediction error (deaths/day)")
abline(h = 0, col = "grey")
```

FIGURE 6.2: *Residuals (vertical axis) vs. the predictor variable of temperature (horizontal axis).*

Plotting Against Another Variable If you have other potential predictor variables, you should be able to plot the residual against them, and also see a flat line around zero. If not, that's an indication that the other variable does in fact help predict the response, and so you should probably incorporate that in your model. (Part of the residuals from your simple linear model are really the contributions of that other predictor, which you were treating as noise out of ignorance[4].) In particular, if you make such a plot and you see the points in it fall around a straight line, that's an excellent sign that you need a multiple linear regression model.

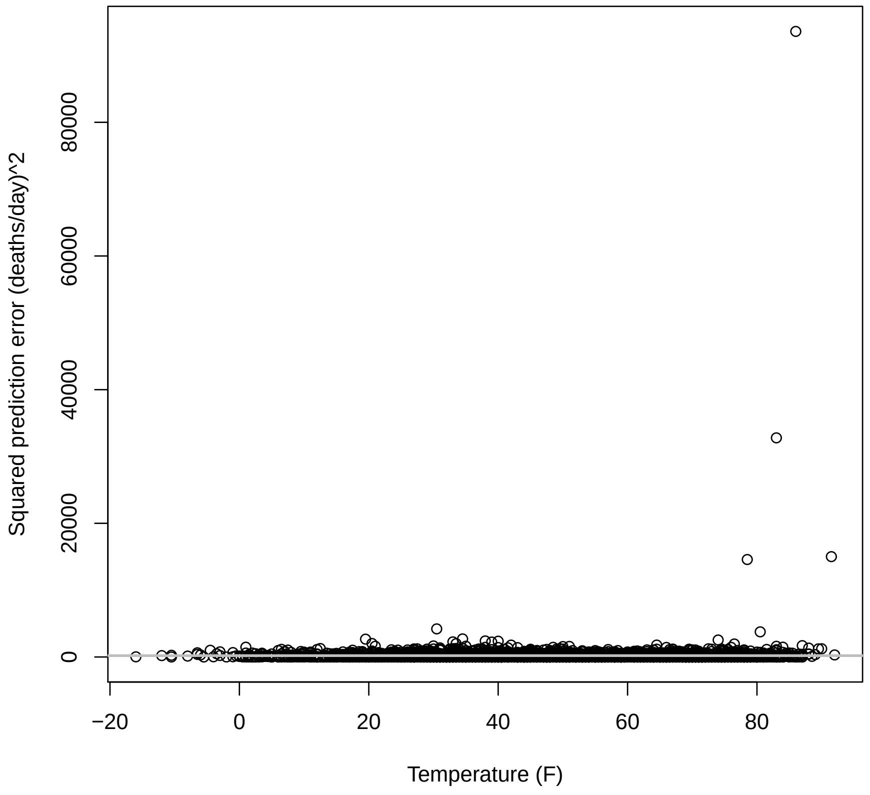
(Exercise: make a plot of the residuals from this model against one of the pollution variables from the data set. Does it look like noise?)

6.1.3 Plot the Magnitude of the Residuals Against the Predictor

Because $\mathbb{E}[e|X=x]=0$, $\text{Var}[e|X=x]=\mathbb{E}\left[e^2|X=x\right]$. (Why?) This means we can check whether the variance of the residuals is constant by plotting the *squared* residuals against the predictor variable. This should give a scatter of points around a flat line, whose height should be around the in-sample MSE. Regions of the x axis where the residuals are persistently above or below this level are evidence of a problem with the simple linear regression model. This could be due to non-constant noise variance ("heteroskedasticity", in the jargon), or due to getting the functional form of the regression wrong. One can often get a clue as to what is driving the problem by looking to see whether the regions where the squared residuals are too big are also regions where the residuals are persistently above or below zero.

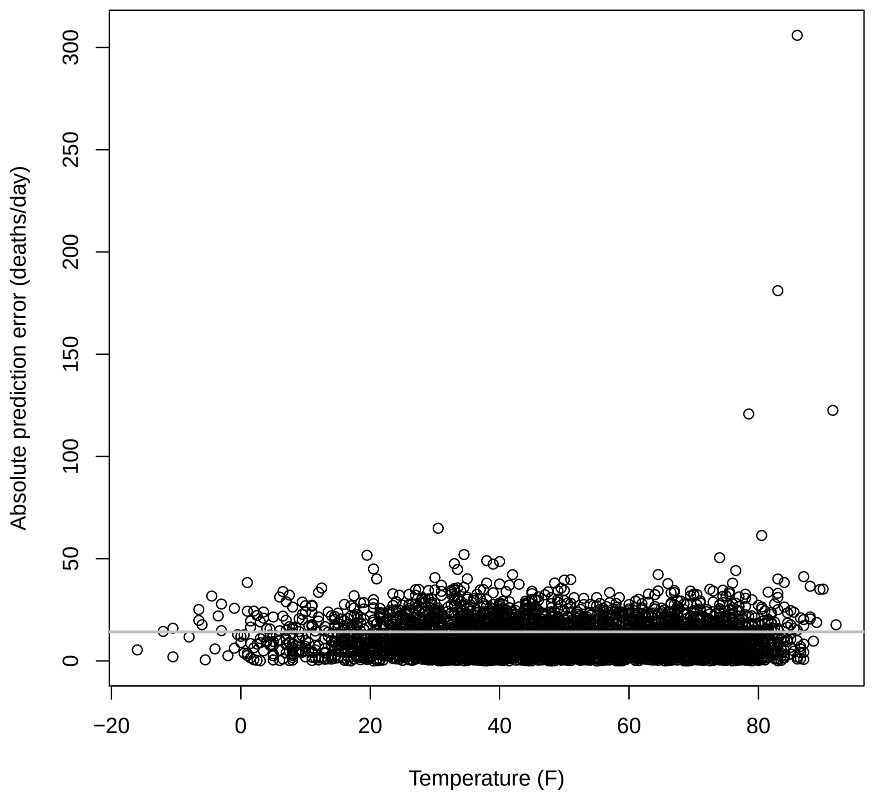
Sometimes, particularly when the model is not doing so well, squaring the residuals leads to a visually uninformative plot, because big residuals lead to really, really big squared residuals, and it's hard to make out any detail. A common fix is to then plot the absolute value of the residuals, with the reference horizontal line being at the square root of the mean squared error.

[4]Whether there really is such a thing as "noise", or only predictor variables we have neglected, is a deep and subtle question we don't have to answer here.



```
# Always plot residuals^2 vs.  predictor variable
plot(chicago$tmpd, residuals(death.temp.lm)^2, xlab = "Temperature (F)", ylab = "Squared prediction error (dea
abline(h = mean(residuals(death.temp.lm)^2), lwd = 2, col = "grey")
```

FIGURE 6.3: *Squared residuals vs. temperature.*



```
plot(chicago$tmpd, abs(residuals(death.temp.lm)), xlab = "Temperature (F)",
    ylab = "Absolute prediction error (deaths/day)")
abline(h = sqrt(mean(residuals(death.temp.lm)^2)), lwd = 2, col = "grey")
```

FIGURE 6.4: *Absolute residuals vs. temperature; plotting these rather than the squared residuals reduces the* visual *impact of the few huge residuals.*

6.1.4 Plot the Residuals Against Coordinates and Each Other

Lots of the time, our data were collected in a certain order — each data point has some coordinates, in space or in time or both. Under the simple linear regression model, these shouldn't matter[5], so you should always plot the residuals against the coordinates. (Even if you have no coordinates, you should always plot residuals against the row numbers of the data set.) Clusters of nearby observations with unusually high or low residuals are a bad sign.

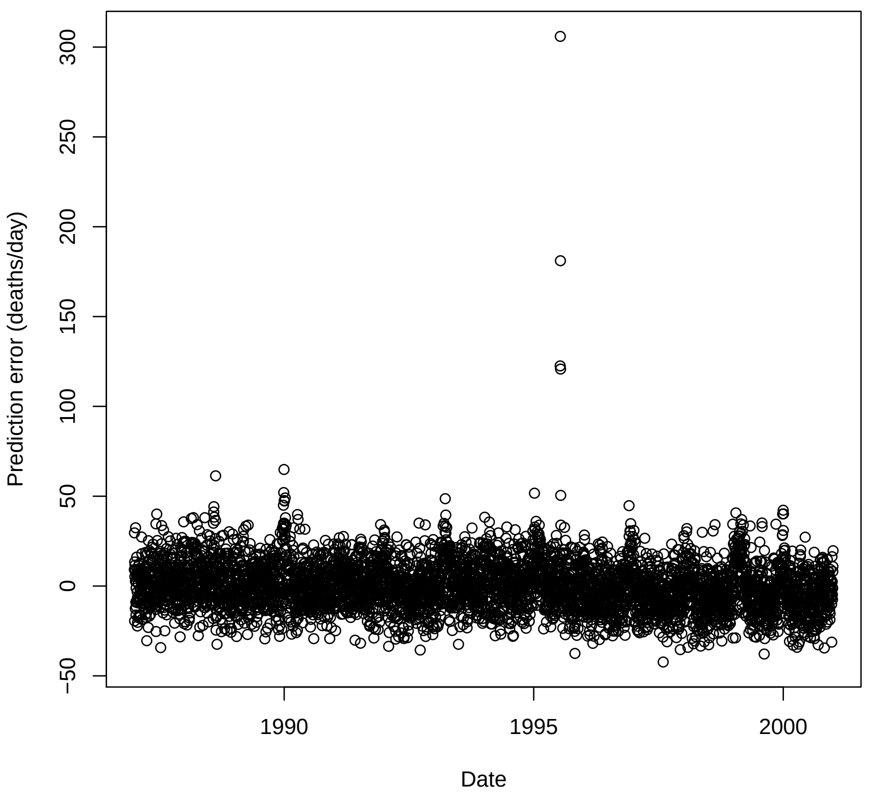
Of course, a certain amount of apparent clustering will happen naturally in any random process, so if this looks worrisome, one should really do some sort of formal test. Fortunately, there are lots of good test procedures for finding "runs" in what should be random noise. A quick hack, though, is simply to put the residuals in a totally random order and re-plot them:

```
sample(residuals(my.model))
```

will take the residuals vector of `my.model` and randomly permute it; plotting these permuted residuals against the coordinate will then give an example of how things should look. Do this a few times, and you'll get a good sense of how much apparent clumping of residuals should be produced by chance. This trick can also be used when plotting the residuals, or squared residuals, against the predictor variable, and can be formalized as what's called a "permutation test".

(Exercise: Make a few plots of the permuted, shuffled residuals against date.)

[5]Unless the predictor variable is one of the coordinates, of course.



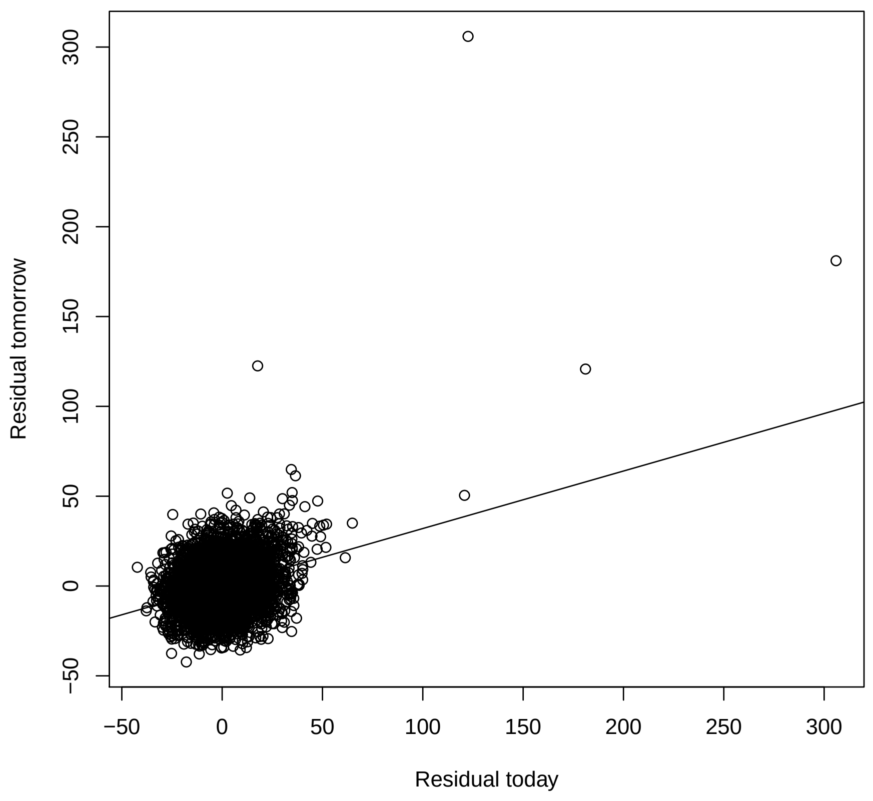
```
# Always plot residuals vs. coordinate
plot(as.Date(chicago$time, origin = "1993-12-31"), residuals(death.temp.lm),
    xlab = "Date", ylab = "Prediction error (deaths/day)")
```

FIGURE 6.5: *Residuals vs. date.*

Residuals vs. Residuals A related diagnostic plot is particularly useful when the observations are taken at regular intervals along some axis (usually time but occasionally distance): make a scatter-plot with one point for each observation except the very last; the horizontal coordinate comes from that point's residual, and the vertical coordinate comes from the *next* point's residual. Ideally, this should give a blob of points with no particular structure. If the residuals are Gaussian, follow any other bell-ish distribution, it should show a circular blob. (If they were uniform, the blob should fill out a square — why?) Falling along a line or curve, or even a tilted ellipse, would be an indication of correlation between successive residuals, which in turn may be a sign that the noise is correlated[6].

Again, if you're not sure whether you're looking at a worryingly big departure from blob-hood, try permuting the residuals before re-plotting.

[6]Or, again, it could be a sign that we've got the functional form wrong, so it's our systematic error which is correlated — see §6.5.



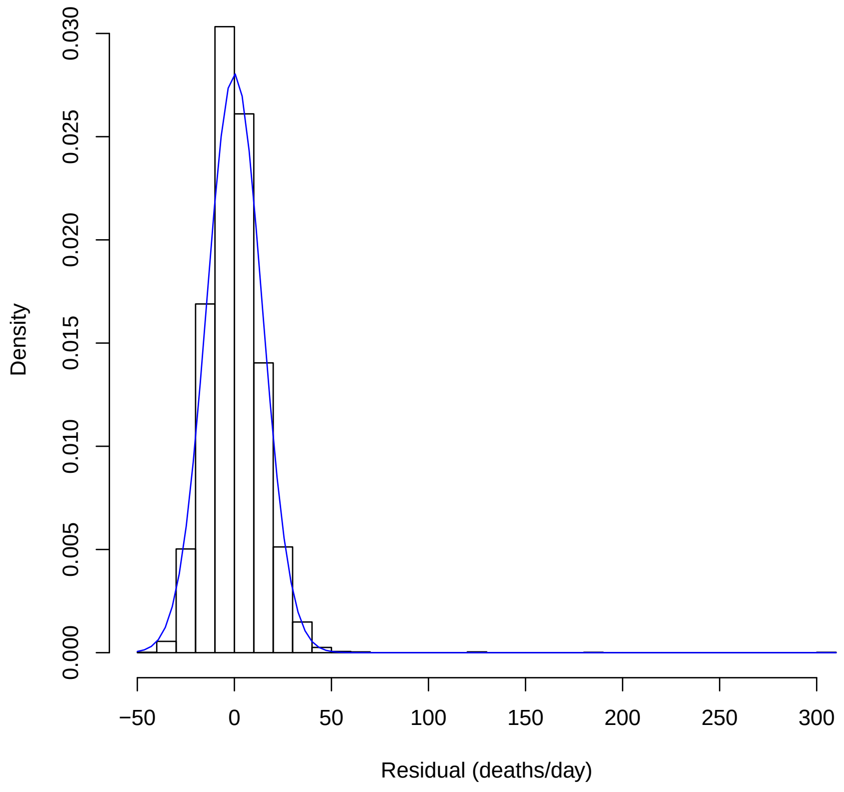
```
# Always plot successive residuals against each other head() and tail() here
# are used to get 'every day except the last' and 'every day except the
# first', respectively see help(head)
plot(head(residuals(death.temp.lm), -1), tail(residuals(death.temp.lm), -1),
    xlab = "Residual today", ylab = "Residual tomorrow")
abline(lm(tail(residuals(death.temp.lm), -1) ~ head(residuals(death.temp.lm),
    -1)))
```

FIGURE 6.6: *Residuals for each day (except the first) plotted as a function of the residuals of the day before. The straight line shows a regression of tomorrow's residual on today's residual, which ideally should be a totally flat line.*

6.1.5 Plot the Distribution of the Residuals

Under the Gaussian noise assumption, the residuals should also follow a Gaussian distribution. We should therefore make plots of the distribution of the residuals, and compare that to a Gaussian.

The most basic plot of the distribution for the residuals is of course a histogram. This should be over-laid with a Gaussian probability density — but which Gaussian? The most reasonable one has mean `0` (because we know the residuals average to `0`), and the same standard deviation as the residuals (because that's the MLE of the standard deviation in a Gaussian model). At that point, one can *see* whether the distribution of residuals looks like that of the best-fitting Gaussian.



```
# Always plot distribution of residuals
hist(residuals(death.temp.lm), breaks = 40, freq = FALSE, xlab = "Residual (deaths/day)",
    main = "")
curve(dnorm(x, mean = 0, sd = sd(residuals(death.temp.lm))), add = TRUE, col = "blue")
```

FIGURE 6.7: *Histogram of the residuals, on a density scale, and the theoretical Gaussian distribution with the same mean (0) and standard deviation.*

Q-Q plots An alternative is what's called a "quantile-quantile" or "Q-Q" plot. (The textbook, old-fashionedly, calls this a "normal probability plot.) This takes a bit more thought to get used to than visually comparing density estimates, but with practice becomes most sensitive and less subjective. Here's the idea.

As you remember from intro. prob., knowing the cumulative distribution function (CDF) tells us all there is to know, mathematically, about a probability distribution; call this $F(x) = \mathbb{P}(X \leq x)$. If the distribution is continuous, the CDF has an inverse function, F^{-1}, where $F^{-1}(p)$ is the unique x such that $\mathbb{P}(X \leq x) = p$. This is called the **quantile function** — it tells us what level of x will contain a fraction p of the total probability. Since saying things like "the 0.95 quantile" is rather awkward, we usually pronounce it as "the 95$^{\mathrm{rm}}$ percentile", meaning the value greater than or equal to 95% of the population. If we know the quantile function, we can invert it to get the CDF, so the quantile function also completely determines the probability distribution[7].

As p varies from 0 to 1, $F^{-1}(p)$ will vary from the smallest possible value for the distribution up to the largest possible value. If our distribution is a Gaussian, with mean μ and variance σ^2, then $F^{-1}(p) = \sigma\Phi^{-1}(p) + \mu$, where Φ is the standard Gaussian CDF. (Why?) So if instead of plotting F^{-1} against p, we make a plot where $F^{-1}(p)$ goes on one axis and $\Phi^{-1}(p)$ goes on the other, as we sweep p from 0 to 1 we'll get a straight line. Conversely, if we weren't sure whether the distribution F we were interested in was Gaussian, but we did one of these quantile-quantile plots against the standard Gaussian and a got a straight line, then we'd know F was, in fact, Gaussian.

With a finite sample from a distribution (like, say, the vector of residuals), we don't really have F or F^{-1}. However, we can use the **sample quantiles** or **empirical quantiles**. Start with our observations, say $x_1, x_2, \ldots x_n$. Now put them in increasing order: to help distinguish the ordered from unordered observations, I'll use a common convention where the subscripts in parentheses show order, so

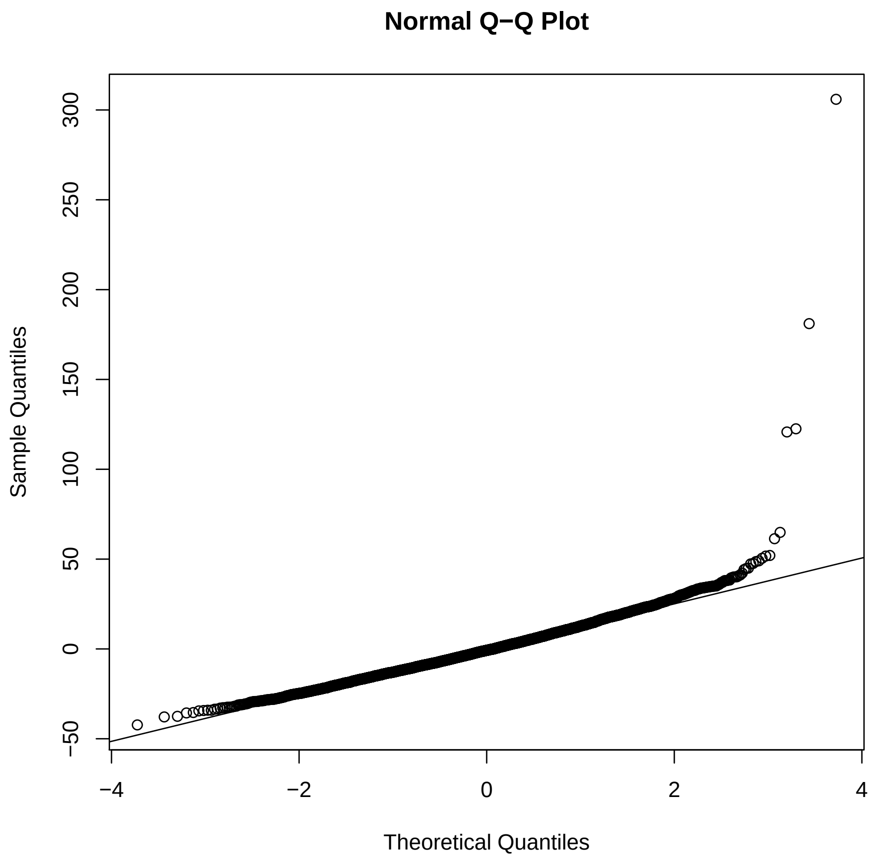
$$x_{(1)} \leq x_{(2)} \leq \ldots \leq x_{(n-1)} \leq x_{(n)} \tag{6.22}$$

(These ordered values of the data are sometimes called the **order statistics**.) Now $x_{(i)}$ is $\geq$ a fraction i/n of the sample observations, so $\hat{F}^{-1}(i/n) = x_{(i)}$. When we make a $Q-Q$ plot against a Gaussian distribution, we therefore put $x_{(i)}$ on one axis, and $\Phi^{-1}(i/n)$ on the other, and hope to see a straight line if the distribution of x is indeed Gaussian[8]

To sum up: we put the data in order, and then plot $x_{(i)}$ against $\Phi^{-1}(i/n)$. If the data are from a Gaussian distribution, these points should fall along a straight line. Small wiggles around the line are to be anticipated from chance; systematic deviations from a line indicate systematic departures from a Gaussian distribution.

[7]In R, the CDF and quantile functions have names beginning with `p` and `q` — `pnorm` and `qnorm` for the Gaussian, `pexp` and `qexp` for the exponential, etc.

[8]You might worry about what happens when $i = n$, since $\Phi^{-1}(1) = \infty$. The answer is that I have slightly over-simplified what goes on that axis. When you call `qqnorm` in R, it actually goes through a calculation to find $\mathbb{E}\left[X_{(i)}\right]$ for $i \in 1:n$ under a Gaussian distribution, and plots *that* against the observed value $x_{(i)}$. As $n \to \infty$, $\mathbb{E}\left[X_{(i)}\right] \to \Phi^{-1}(i/n)$, but the difference can be important for small n, and/or i/n close to 0 or 1.



```
# An alternative: plot vs. theoretical Gaussian distribution
qqnorm(residuals(death.temp.lm))
qqline(residuals(death.temp.lm))
```

FIGURE 6.8: *QQ plot of the residuals, using the standard Gaussian distribution as the reference distribution.*

Q-Q plots for other distributions One can make a Q-Q plot for data against any other distribution one likes, provided one knows the reference distribution's quantile function; R just provides code for the Gaussian case, however.

Q-Q plots for two data distributions With two data sets, say $x_1, \ldots x_n$ and y_1, y_m, one can compare their sample quantiles. The easiest way is when $n = m$, since then one just plots $x_{(i)}$ against $y_{(i)}$. (If $n \neq m$, one might pick a grid of p values, work out $\hat{F}_x^{-1}(p)$ and $\hat{F}_y^{-1}(p)$ by interpolation, and plot those against each other.) This should show points around the $x = y$ line when x and y are both drawn from the same distribution.

P-P plots An alternative to $Q - Q$ plots are $P - P$ plots, where both axes run from 0 to 1. Call the horizontal coordinate p_1 and the vertical coordinate p_2; what we plot is $p_2 = \hat{F}(F^{-1}(p_1))$. $F^{-1}(p_1)$ is the quantile which ought, under the distribution F, to be $\geq$ a fraction p_1 of the population; $\hat{F}(F^{-1}(p_1))$ shows the actual fraction of the sample which it exceeds. R doesn't have a built-in function to make this — can you write one?

Formal tests Comparing the histogram to a theoretical density can be formalized with a χ^2 test[9]. Checking whether the $Q - Q$ plot follows a straight line can be formalized in the Kolmogorov-Smirnov test. In both cases, since we're really estimating the a parameter of the reference distribution (the standard deviation), we need to take some care to account that in a hypothesis test. (A tweak to the K-S test which does this, when testing for Gaussianity, is the "Lilliefors test", which you can find in the package `nortest`.)

[9]This is why the visual comparison is informally called "χ by eye".

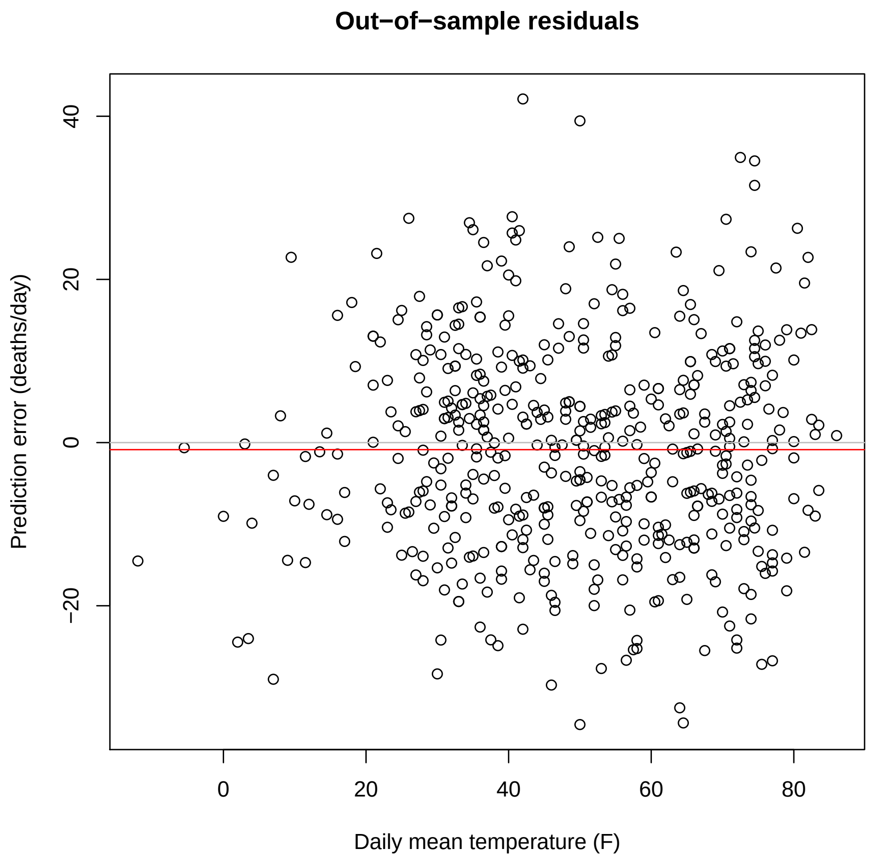
```
# Always look at whether the model can extrapolate to new data Basic check:
# randomly divide into two parts, here say 90% of the data vs. 10% Use the
# 'training set' to estimate the model
training.rows <- sample(1:nrow(chicago), size = round(nrow(chicago) * 0.9),
    replace = FALSE)
training.set <- chicago[training.rows, ]
# We'll use the 'testing set' to see how well it does
testing.set <- chicago[-training.rows, ]
# Estimate the model on the training set only
training.lm <- lm(death ~ tmpd, data = training.set)
# Make predictions on the testing set The model didn't get to see these
# points while it was being estimated, so this really checks (or tests)
# whether it can predict
testing.preds <- predict(training.lm, newdata = testing.set)
# Unfortunately residuals() doesn't know about the new data set so calculate
# the residuals by hand
testing.residuals <- testing.set$death - testing.preds
```

FIGURE 6.9: *Code setting up a* random *division of the data into training and testing sets, and looking at how well the model does on points in the testing set (which it didn't get to see during estimation).*

6.1.6 Generalization

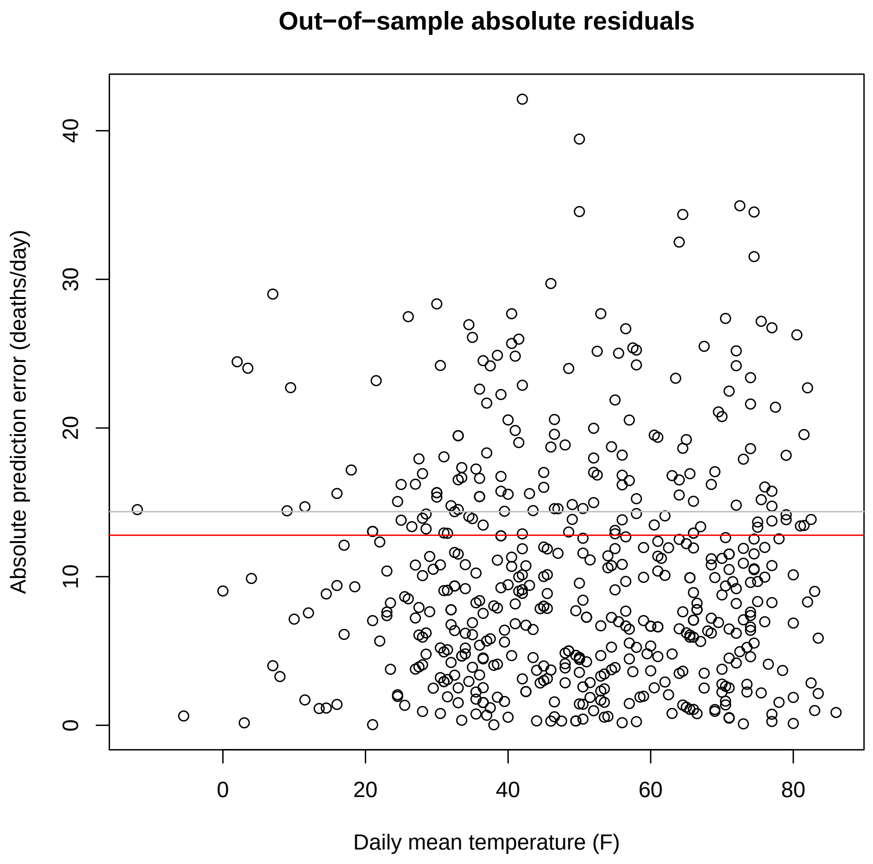
If the model assumptions are correct, it should be able to work about equally well on new data from the same source. Because the parameters were adjusted to fit the data we used to estimate the model, we should expect the prediction errors on new data to be slightly larger in magnitude, but they shouldn't be biased or otherwise show patterns. An important basic check on the model is therefore to divide the data into two parts, estimate the model on one part, the **training set**, and then examine the predictions and the residuals on the rest of the data, the **testing set**.

We can either make the division into training and testing sets by random sampling, or systematically. A random division ensures that the testing set has (almost) the same distribution as the training set. The averaged squared error on the testing set is therefore an unbiased estimate of the true mean squared error on new data. We will topic later in the course, under the heading of "cross-validation", since it is one of the most useful ways of selecting among competing models.



```
# Plot our residuals against the predictor variable
plot(testing.set$tmpd, testing.residuals, xlab = "Daily mean temperature (F)",
    ylab = "Prediction error (deaths/day)", main = "Out-of-sample residuals")
abline(h = 0, col = "grey")
abline(h = mean(testing.residuals), col = "red")
```

FIGURE 6.10: *Plot of residuals vs. temperature for the testing set. Remember that the data points here were* not *available to the model during estimation. The grey line marks the average we'd see on the training set (zero), while the red line shows the average on the testing set.*



```
# Plot absolute residuals vs. predictor variable
plot(testing.set$tmpd, abs(testing.residuals), xlab = "Daily mean temperature (F)",
    ylab = "Absolute prediction error (deaths/day)", main = "Out-of-sample absolute residuals")
abline(h = sqrt(mean(residuals(training.lm)^2)), col = "grey")
abline(h = sqrt(mean(testing.residuals^2)), col = "red")
```

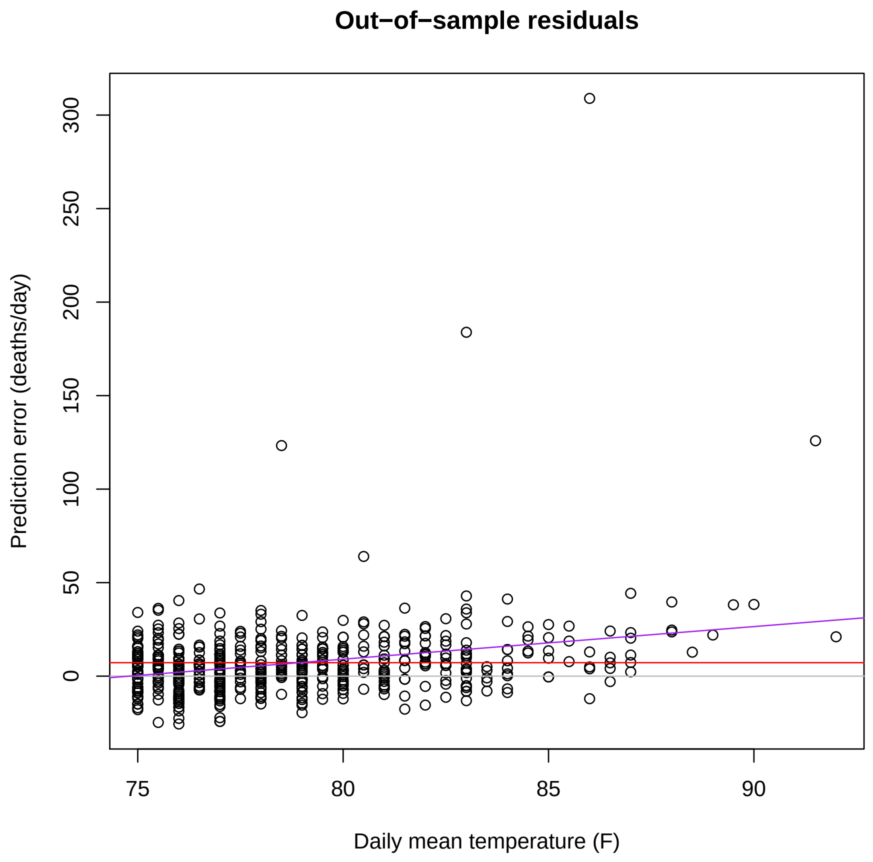
FIGURE 6.11: *As in Figure 6.10, but looking at the squared residuals.*

```
# Find the low-temperature days
lowtemp.rows <- which(chicago$tmpd < 75)  # About 90% of the data
# Divide into low- and high- temperature data sets
lowtemp.set <- chicago[lowtemp.rows, ]
hightemp.set <- chicago[-lowtemp.rows, ]
# Estimate the model on the colder days only
lowtemp.lm <- lm(death ~ tmpd, data = lowtemp.set)
# For you: how much do the parameters change, as compared to using all the
# data?  Now predict on the high-temperature days Again, these are new data
# points, but now systematically different (because of their temperature)
# from the data used to estimate
hightemp.preds <- predict(lowtemp.lm, newdata = hightemp.set)
# Calculate our own residuals
hightemp.residuals <- hightemp.set$death - hightemp.preds
```

FIGURE 6.12: *Setting up a division of the data into a low-temperature training set and a high-temperature testing set.*

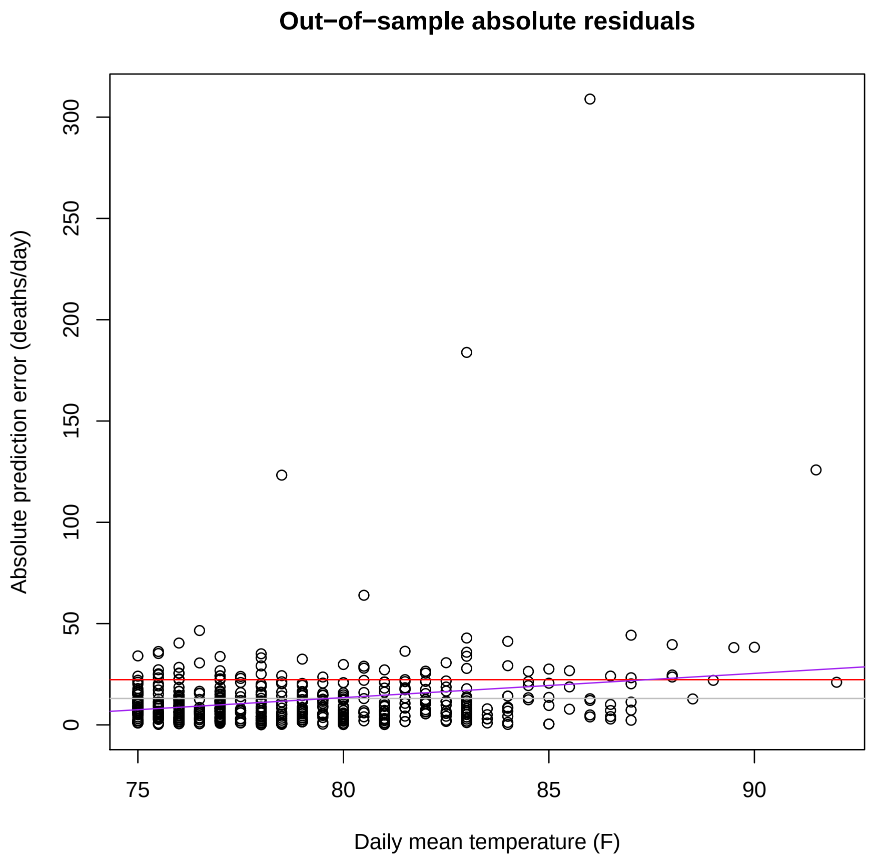
Extrapolative Generalization An alternative to random division, which can be even more useful for model checking, is to systematically make the testing set into a part of the data where the over-all fit of the model seems dubious based on other diagnostics. With our running examples, for instance, the model seems to do decently at lower temperatures, but starts to look iffy at high temperatures. We might, then, fit the model to low temperatures, and see whether it can extrapolate to high temperatures, or whether it seems to make systematic errors there. (We could also estimate the model on the high-temperature portion of the data, and see how well it extrapolates to the lower temperatures, or estimate on the middle range and see about both extremes, etc., etc.)

Generalize All the Things! *All* of the diagnostic plots discussed earlier can be combined with the trick of estimating the model on one part of the data, and then seeing whether it generalizes to the testing set. This is slightly more complicated than doing everything on the full data, but arguably has more power to detect problems with the model.



```
# Plot residuals vs. temperature
plot(hightemp.set$tmpd, hightemp.residuals, xlab = "Daily mean temperature (F)",
    ylab = "Prediction error (deaths/day)", main = "Out-of-sample residuals")
# Flat line at 0 (ideal, if the model is right)
abline(h = 0, col = "grey")
# Flat line at the mean of the new residuals
abline(h = mean(hightemp.residuals), col = "red")
# Regressing the new residuals on temperature does not look good...
abline(lm(hightemp.residuals ~ hightemp.set$tmpd), col = "purple")
```

FIGURE 6.13: *Residuals vs. temperature, where the testing set here consists of days with temperature ≥ 75 degrees Farenheit, and the training set only those < 75 degrees. The grey line indicates the average residual we'd see on the training data (zero); the red line the average residual on the testing data; the purple a regression of residual on temperature, which ideally should have had slope zero.*



```
# Similar plots for the absolute residuals
plot(hightemp.set$tmpd, abs(hightemp.residuals), xlab = "Daily mean temperature (F)",
    ylab = "Absolute prediction error (deaths/day)", main = "Out-of-sample absolute residuals")
abline(h = sqrt(mean(residuals(lowtemp.lm)^2)), col = "grey")
abline(h = sqrt(mean(hightemp.residuals^2)), col = "red")
abline(lm(abs(hightemp.residuals) ~ hightemp.set$tmpd), col = "purple")
```

FIGURE 6.14: *As in Figure 6.13, but for absolute residuals.*

6.2 Nonlinear Functions of X

When we plot the residuals against the predictor variable, we may see a curved or stepped pattern. This strongly suggests that the relationship between Y and X is not linear. At this point, it is often useful to fall back to, in fact, plotting the y_i against the x_i, and try to guess at the functional form of the curve.

6.2.1 Transformations

The easy case is when $Y = \beta_0 + \beta_1 f(x) + \epsilon$ for some fixed, easily-computed function f like $\sqrt{x}$, x^2, $\log x$, $\sin x$, etc. We then calculate $f(x_i)$, and run a simple linear regression of y_i on $f(x_i)$. The interpretation of the coefficients hardly changes, except that we need to replace X everywhere with $f(X) - \beta_0 = \mathbb{E}[Y|f(X) = 0]$, β_1 is the difference $\mathbb{E}[Y|f(X) = f_0]$ and $\mathbb{E}[Y|f(X) = f_0 - 1]$, etc. This is called "transforming the predictor", and it only works this simply when the transformation itself doesn't have to be estimated, but can just be guessed at. Ideally, in fact, we derive the transformation from some physical (chemical, biological, psychological, sociological, economic, ...) theory. For instance, there are good reasons in physiology and psychology to say that an organism's behavioral response to a stimulus should vary with the logarithm of the stimulus's physical intensity[10]. A good check on such a transformation is to plot the y_i against the $f(x_i)$, and see that the data now fall on a straight line.

6.2.2 Nonlinear Least Squares

If the transformation does have to be estimated, but the functional form is known[11], then the method of least squares (or maximum likelihood) still applies. Taking the derivative of the mean squared error (or the log likelihood) with respect to the parameters and setting them equal to zero gives a set of normal or estimating equations. Usually, however, these equations are nonlinear, and don't have a closed-form solution. Finding the solution is thus called "solving a nonlinear least squares (NLS) problem". When we have theoretical reasons to use some thoroughly nonlinear model, say $Y = \beta_0 x^{\beta_1} + \epsilon$, we can still estimate it using NLS in this way. Aside from needing to solve the estimating equations numerically, there are some special considerations to NLS, which we'll either cover later in this class (time permitting) or in 402[12].

6.2.3 Smoothing

An alternative to using a parametrized nonlinear model is to try to let the data tell us what the appropriate curve is — to take a "non-parametric" approach. The most

[10]This is also (roughly) true, at least in people, of the perceived intensity of the stimulus. Full sunlight carries $\approx 1000\text{W}/\text{m}^2$ of power, while the light of a full moon is $\approx 0.025\text{W}/\text{m}^2$, yet moonlight doesn't seem forty thousand times dimmer. The logarithmic relationship between perceived and physical intensity is sometimes called "Fechner's law" in psychophysics.

[11]For instance, if $Y = \beta_0 + \beta_1 \log(x + \beta_2) + \epsilon$, for unknown k.

[12]The relevant R commands are `optim`, which is a general-purpose minimization function, and `nls`, which is, unsurprisingly, specialized to nonlinear least squares.

basic sort of curve-fitting would just take a little interval around any point x, say from $x-h$ to $x+h$, and average all the y_i where x_i was in the interval:

$$\hat{m}(x) = \frac{\sum_{i=1}^{n} y_i I_{[x-h,x+h]}(x_i)}{\sum_{j=1}^{n} I_{[x-h,x+h]}(x_j)} \tag{6.23}$$

(Here $I_{[a,b]]}(x)$ is the **indicator function** for the interval $[a,b]$, i.e., 1 if $a \leq x \leq b$, and `0` otherwise.)

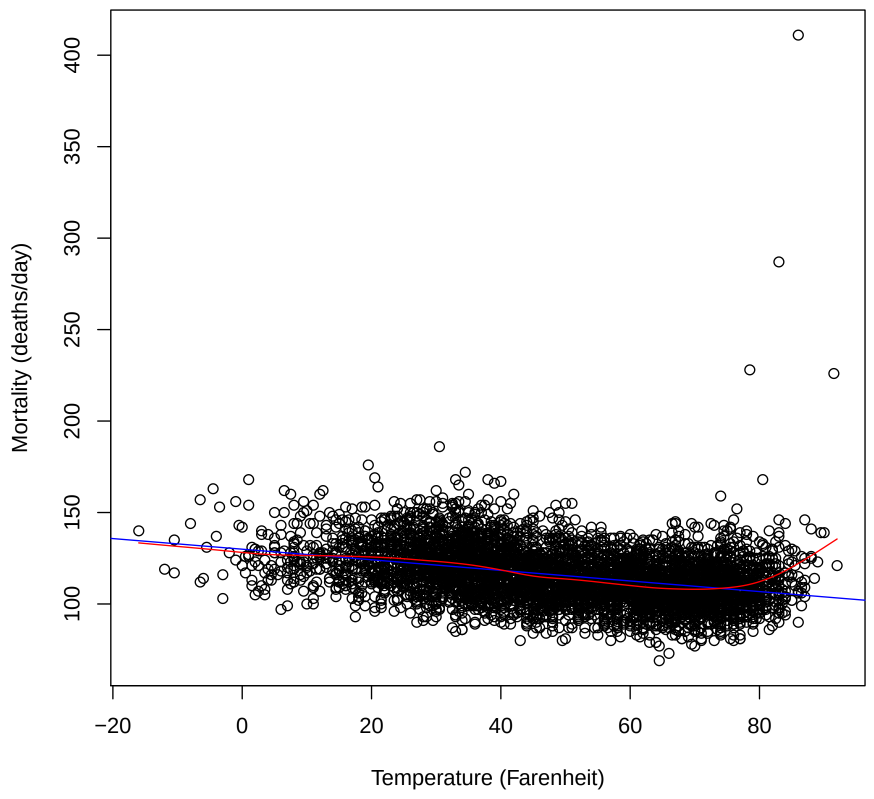
This sort of local averaging gives an $\hat{m}(x)$ which can make jerk steps as x changes. Another approach is **spline smoothing**: this looks for the function, called a **spline** which comes closest to the data points, subject to a constraint on the average curvature of the function. Allowing no curvature at all gives back the least squares line; allowing unlimited curvature gives a function which interpolates exactly between the data points[13]. Of course one has to decide how much curvature to allow; the best idea is generally to do what's called "cross-validation": hold back a little bit of the data, fit the spline with some level of curvature to the rest of the data, and see how well it predicts the held-back part; pick the curvature which generalizes best to unseen data points. While there is lot of R code for splines, because they're very useful, the most user-friendly is called `smooth.spline`.

```
smooth.spline(x, y, cv = TRUE)
```

returns an object which contains the estimated spline. (The default, `CV=FALSE`, is to use a fast approximation to cross-validation called "generalized cross-validation"; `CV=TRUE` is often a bit more accurate, if you can afford the time.) The commands `fitted` and `residuals` work on this object just like they do on `lm` objects; `predict` works very similarly, but the argument giving the new values must be a vector called `x`, not a data frame. If passed into the `lines` or `points` command, we get a plot of the fitted values.

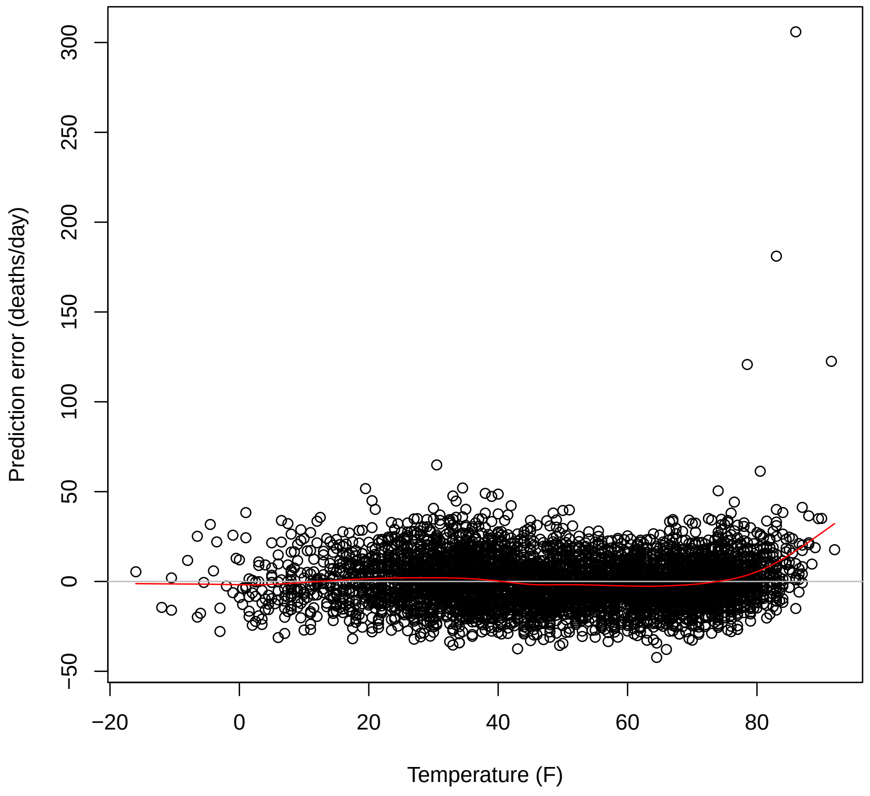
`smooth.spline` can also be used on the squared residuals (not the absolute residuals – why?) to try to get an estimate of the conditional variance, if you're pretty sure that the functional form of the regression is right.

[13]This turns out to be equivalent to a kind of weighted local averaging.



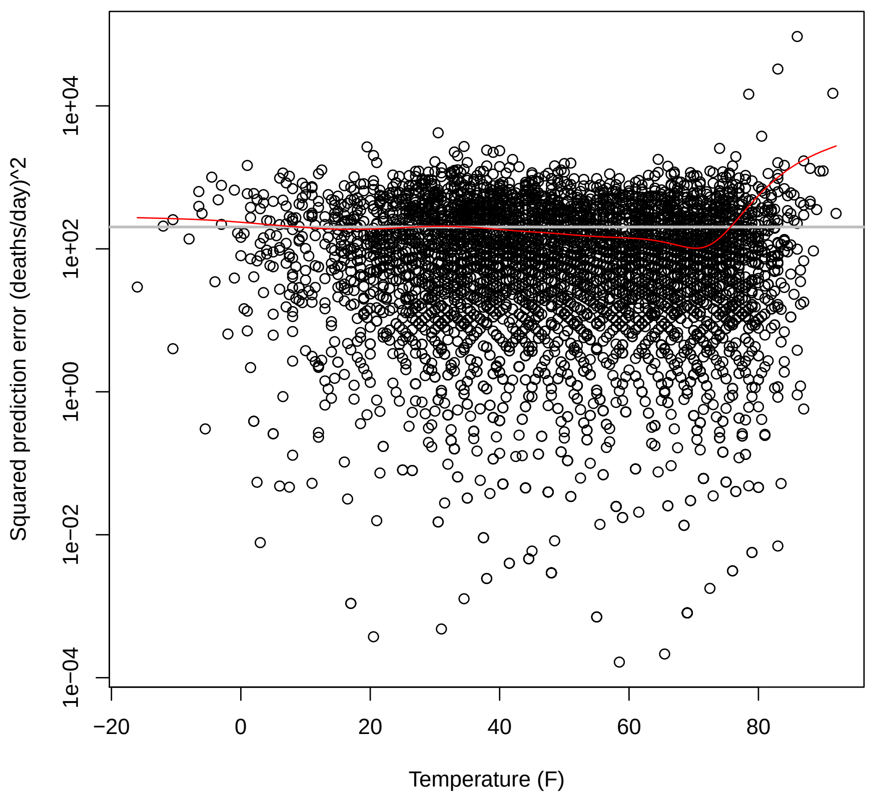
```
plot(death ~ tmpd, data = chicago, xlab = "Temperature (Farenheit)", ylab = "Mortality (deaths/day)
death.temp.ss <- smooth.spline(x = chicago$tmpd, y = chicago$death, cv = TRUE)
abline(death.temp.lm, col = "blue")
lines(death.temp.ss, col = "red")
```

FIGURE 6.15: *Scatter-plot of mortality as a function of temperature, along with the estimated linear model (blue) and the estimated smoothing spline (red). Notice how the smoothing spline tracks the linear model over a wide range of temperatures, but then turns* up *for high temperatures.*



```
plot(chicago$tmpd, residuals(death.temp.lm), xlab = "Temperature (F)", ylab = "Prediction error (deaths/day)")
abline(h = 0, col = "grey")
lines(smooth.spline(x = chicago$tmpd, y = residuals(death.temp.lm), cv = TRUE),
    col = "red")
```

FIGURE 6.16: *Estimating a smoothing spline on the residuals of the linear model. Ideally, the spline would be close to zero everywhere; here we see it's working pretty well,* except *at high temperature.*



```
plot(chicago$tmpd, residuals(death.temp.lm)^2, log = "y", xlab = "Temperature (F)",
    ylab = "Squared prediction error (deaths/day)^2")
abline(h = mean(residuals(death.temp.lm)^2), lwd = 2, col = "grey")
lines(smooth.spline(x = chicago$tmpd, y = residuals(death.temp.lm)^2, cv = TRUE),
    col = "red")
```

FIGURE 6.17: *Smoothing spline of squared residuals versus temperature. (Notice the logarithmic scale for the vertical axis, to compensate for the fact that some of the residuals are really big.) If we thought that the functional form of the regression was right, this would be a reasonable estimate of the conditional variance. Should we think that?*

6.3 Transforming the Response

Another way to try to accommodate nonlinearity is to transform the response variable, rather than the predictor. That is, one imagines the model is

$$g(Y) = \beta_0 + \beta_1 x + \epsilon_i \tag{6.24}$$

for some invertible function g. In more old-fashioned sources, like our textbook, this is advocated as a way of handling non-constant variance, or non-Gaussian noise[14]. A better rationale is that it might in fact be true. Since the transformation g has an inverse, we can write

$$Y = g^{-1}(\beta_0 + \beta_1 x + \epsilon_i) \tag{6.25}$$

Even if $\epsilon_i \sim N(0, \sigma^2)$, this implies hat Y will have a non-Gaussian distribution, with a non-linear relationship between $\mathbb{E}[Y|X = x]$ and x, and a non-constant variance. If that's actually the case, we'd like to incorporate that into the model. For instance, Y might be an integer-valued count variable, or even a binary-valued categorical variable, and then we pretty much have to have some sort of non-Gaussian noise in Y. (Indeed, the noise around $\mathbb{E}[Y|X = x]$ isn't even strictly additive.)

Let me illustrate these points by working through a log transformation of Y. Suppose

$$\log Y = \beta_0 + \beta_1 x + \epsilon \tag{6.26}$$

where $\epsilon \sim N(0, \sigma^2)$, independent of x. The inverse function to log is exp, so this is logically equivalent to

$$Y = e^{\beta_0 + \beta_1 x} e^{\epsilon} = e^{\beta_0 + \beta_1 x} \xi \tag{6.27}$$

where $\xi = e^{\epsilon}$ follows a **log-normal** distribution, with $\mathbb{E}[\log \xi] = 0$, $\mathrm{Var}[\log \xi] = \sigma^2$. One can show[15] that $\mathbb{E}[\xi] = e^{\sigma^2/2}$, that $\mathrm{Var}[\xi] = (e^{\sigma^2} - 1)e^{\sigma^2}$, that the median of ξ is 1, and that ξ is, consequently, skewed to the right, with an asymmetric distribution. Conversely, if $Y = e^{\beta_0 + \beta_1 x} + \epsilon$, with $\epsilon \sim N(0, \sigma^2)$, then it is *not* the case that $\log Y = \beta_0 + \beta_1 x + \eta$, for some Gaussian η.

The point of this is that transforming the response thoroughly changes the interpretation of the parameters: β_0 is not $\mathbb{E}[Y|X = x]$, but $\mathbb{E}[g(Y)|X = x]$, β_1 is the slope of $\mathbb{E}[g(Y)|X = x]$, etc. It also implies very particular, even peculiar, models for noise around the regression line. This makes it impossible to compare mean squared errors or log-likelihoods before and after transformations[16]. One can, however, look at the residuals for the *transformed* response, and see whether they are flat in the predictor variable, etc.

6.3.1 Box-Cox Transformations

The great statisticians G. E. P. Box and D. R. Cox introduced a family of transformations which includes powers of Y and taking the logarithm of Y, parameterized by a

[14]To be quite honest, this seems like as clear a case of being enslaved by one's own tools as one could hope to find in the sciences. More charitably: it made a certain amount of sense before people figures out how to deal with non-constant variance (around 1935) and non-Gaussian noise (around 1980).

[15]Check Wikipedia if you don't believe me.

[16]More exactly, log-likelihoods *can* be compared, but we need to compensate for the transformation.

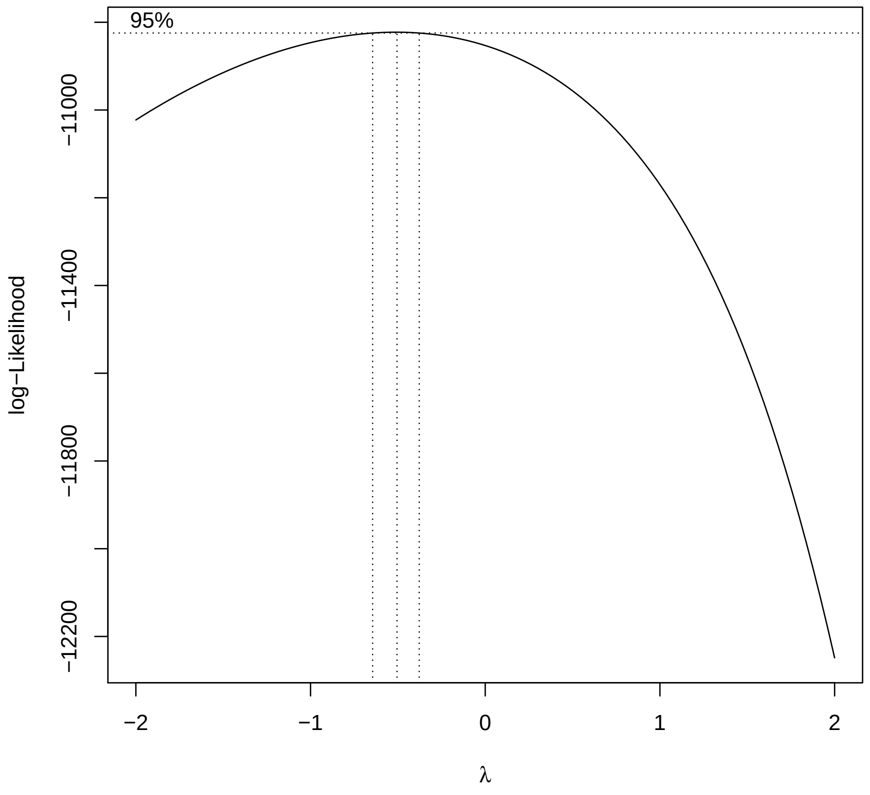
number λ:

$$b_\lambda(y) = \frac{y^\lambda - 1}{\lambda} \tag{6.28}$$

In the limit $\lambda \rightarrow 0$, this becomes $\log y$ (Exercise 2). If one assumes that

$$\begin{aligned}
b_\lambda(Y) &= \beta_0 + \beta_1 x + \epsilon && (6.29)\\
\epsilon &\sim N(0, \sigma^2) && (6.30)\\
\epsilon &\ \text{independent of}\ x && (6.31)\\
& && (6.32)
\end{aligned}$$

then one can estimate λ by maximizing the likelihood. This is implemented in R through the function `boxcox` in the package `MASS`.

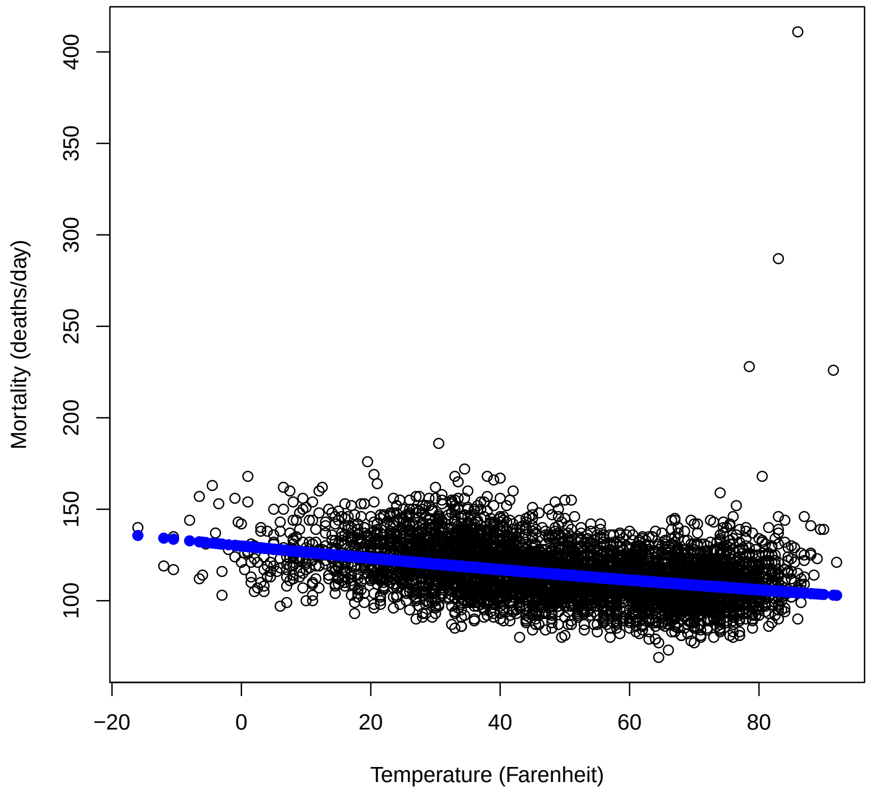


```
library(MASS)
# Works with a previously-fit lm model
boxcox(death.temp.lm)
```

FIGURE 6.18: *Plot of the log-likelihood of different values of λ in the Box-Cox transformation, applied to the regression of deaths on temperature. The dashed lines indicate an approximate 95% confidence interval for λ.*

```
# You can also give a model formula, and suppress plotting
bc.death.temp <- boxcox(death ~ tmpd, data = chicago, plotit = FALSE)
# Result is a list showing lambda values ($x component) vs. log-likelihood
# (the $y component) We can use this to get a (rough) maximum value for
# lambda
(lambda.hat <- bc.death.temp$x[which.max(bc.death.temp$y)])
## [1] -0.5
```

FIGURE 6.19: *Another way of using the* `boxcox` *function.*



```
chicago$bc.death <- (chicago$death^lambda.hat - 1)/lambda.hat
bc.lm <- lm(bc.death ~ tmpd, data = chicago)
plot(death ~ tmpd, data = chicago, xlab = "Temperature (Farenheit)", ylab = "Mortality (deaths/day)")
points(chicago[, "tmpd"], (lambda.hat * fitted(bc.lm) + 1)^(1/lambda.hat), pch = 19,
    col = "blue")
```

FIGURE 6.20: *Actual data (hollow circles) and fitted values from the Box-Cox transformation.*

It is important to remember that this family of transformations is just something that Box and Cox made up because it led to tractable math. There is absolutely no justification for it in probability theory, general considerations of mathematical modeling, or scientific theories. There is also no general reason to think that the correct model will be one where some transformation of Y is a linear function of the predictor variable plus Gaussian noise. Estimating a Box-Cox transformation by maximum likelihood does not relieve us of the need to run all the diagnostic checks *after* the transformation. Even the best Box-Cox transformation may be utter rubbish.

I will add that while Box-Cox transformations have been part of courses like this since they were introduced in 1964, I have never encountered a real, non-textbook data-analysis problem where they helped, with *one* exception, which I will discuss immediately.

Log transformations with log-normal multiplicative noise The usual argument for anticipating Gaussian noise is the central limit theorem: if lots of little disturbing causes add up their contributions to the response variable, and they're roughly independent and of comparable size, we'll expect that their net effect is Gaussian. If instead of adding up they multiply together, however, the central limit theorem does not apply directly. Since logarithms turn multiplication into addition, when effects multiply we may anticipate log-normal fluctuations. If the simple linear regression model applies after taking the log of everything,

$$\log Y = \beta_0 + \beta_1 log x + \epsilon \quad (6.33)$$

$$\epsilon \sim N(0, \sigma^2) \quad (6.34)$$

$$\epsilon \text{ independent of } X \quad (6.35)$$

then, undoing the log,

$$Y = e^{\beta_0} x^{\beta_1} \eta \quad (6.36)$$

$$\eta \sim \log N(0, \sigma^2)$$

$$\eta \text{ independent of } X$$

This sort of model is not actually unheard of in practice, because there really are situations where causes multiply.

Eq. 6.36 is a *different model* from

$$Y = e^{\beta_0} x^{\beta_1} + \epsilon \quad (6.37)$$

We would estimate the former model by doing a linear regression of $\log Y$ on $\log X$ with Gaussian noise. We would estimate the latter model by nonlinear least squares. The two estimates will not, in general, agree.

6.4 Looking Forward

If you need more exact formulas for the variance and correlation of the residuals, they are most easily derived from the matrix approach to linear regression, as gone over in Chapter 11 (specifically §11.2.1).

Non-constant variance in a linear model is actually comparatively easy to handle, if we can work out what variance is: the trick is a modification of the method of least squares called "weighted least squares", which we will cover later in the course. Similarly, correlated noise can be handled through a modification called "generalized least squares", which we'll also cover. There are a range of simulation-based techniques for doing inference when the Gaussian-noise assumption fails; these have opaque, forcedly-whimsical names like "the jackknife" and "the bootstrap", and we'll get to these towards the end of the course. Dealing with nonlinearity will occupy us for much of 402, though in a few weeks we will look at polynomial regression.

The easiest direction to go in is to add more predictor variables, and hope that the response is linear in each of them. We will explore this path after the midterm.

6.5 Residuals under Misspecification

(This section is optional, but strongly recommended.)

Suppose that the conditional expectation function $\mu(x) \equiv \mathbb{E}[Y|X=x]$ is not linear in x, but we use a linear model anyway. We know, from Chapter 1, that there is a best linear approximation to this μ, with slope $\beta_1 = \mathrm{Cov}[X,Y]/\mathrm{Var}[X]$ and intercept $\beta_0 = \mathbb{E}[Y] - \beta_1 \mathbb{E}[X]$. So we can write

$$\begin{aligned} Y &= \mu(x) + \epsilon && (6.38) \\ Y &= \beta_0 + \beta_1 x + (\mu(x) - \beta_0 + \beta_1 x) + \epsilon && (6.39) \end{aligned}$$

It's annoying to keep writing out $\mu(x) - \beta_0 + \beta_1 x$, so let's define that to be a new function, $\nu(x) \equiv \mu(x) - \beta_0 + \beta_1 x$. You can show (Exercise 3) that

$$\mathbb{E}[\nu(X)] = 0 \tag{6.40}$$

and that

$$\mathrm{Cov}[\nu(X), X] = 0 \tag{6.41}$$

However, it is not in general the case that

$$\mathbb{E}[\nu(X)|X=x] = 0 \tag{6.42}$$

Suppose now that — through ignorance or as a deliberate strategy — we fit a linear model by least squares. It will be the case that

$$Y_i = \beta_0 + \beta_1 x_i + \eta_i \tag{6.43}$$

where η_i has expectation zero and is uncorrelated with X. But what we are treating as noise is partially the real noise, and partially the systematic effect of ignoring the nonlinearities:

$$\eta_i = \epsilon_i + \nu(x_i) \tag{6.44}$$

Thus, what looks like noise to the linear model will *not* have conditional expectation zero:

$$\mathbb{E}[\eta|X=x] = \nu(x) \tag{6.45}$$

The conditional variance doesn't alter,

$$\mathrm{Var}[\eta|X=x] = \mathrm{Var}[\epsilon|X=x] \tag{6.46}$$

but it's no longer equal to the expected square:

$$\mathbb{E}\left[\eta^2|X=x\right] = \nu^2(x) + \mathrm{Var}[\epsilon|X=x] \tag{6.47}$$

As mentioned in earlier chapters, when we run least squares, our slope and intercept will still tend to converge on those of the optimal linear model. We can even still use the expressions we worked out for $\hat{\beta}_0$ (or $\hat{\beta}_1$) in terms of β_0 (or β_1) and a weighted sum of random variables — but now those are the η_i, not the ϵ_i. This implies that conditional expectation of the residuals will *not* be zero (in general), the conditional expectation of the squared residuals will not be constant, and so forth.

Exercises

1. Using Eq. 6.9, find the covariance, conditional on all the x_i, between e_i and e_k, for arbitrary $i, k \in 1:n$.
2. Show that $b_\lambda(y) \rightarrow \log y$ as $\lambda \rightarrow 0$. *Hint:* L'Hôpital's rule.
3. Consider the $\nu(x)$ function from §6.5.
   (a) Show that $\mathbb{E}[\nu(X)] = 0$. *Hint:* Show that $\mathbb{E}[Y] = \mathbb{E}[\mu(X)]$.
   (b) Show that $\mathrm{Cov}[X, \nu(X)] = 0$. *Hint:* Show that $\mathrm{Cov}[X, Y] = \mathrm{Cov}[X, \mu(X)]$.

Chapter 7

Inference on Parameters

Having gone over the Gaussian-noise simple linear regression model, over ways of estimating its parameters and some of the properties of the model, and over how to check the model's assumptions, we are now ready to begin doing some serious statistical inference within the model[1]. In previous chapters, we came up with **point estimators** of the parameters and the conditional mean (prediction) function, but we weren't able to say much about the margin of uncertainty around these estimates. In this chapter we will focus on supplementing point estimates with *reliable* measures of uncertainty. This will naturally lead us to testing hypotheses about the true parameters — again, we will want hypothesis tests which are unlikely to get the answer wrong, whatever the truth might be.

To accomplish all this, we first need to understand the sampling distribution of our point estimators. We can find them, mathematically, but they involve the unknown true parameters in inconvenient ways. We will therefore work to find combinations of our estimators and the true parameters with fixed, parameter-free distributions; we'll get our confidence sets and our hypothesis tests from them.

Throughout this chapter, I am assuming, unless otherwise noted, that all of the assumptions of the Gaussian-noise simple linear regression model hold. After all, we checked those assumptions last time....

7.1 Sampling Distribution of $\hat{\beta}_0$, $\hat{\beta}_1$ and $\hat{\sigma}^2$

The Gaussian-noise simple linear regression model has three parameters: the intercept β_0, the slope β_1, and the noise variance σ^2. We've seen, previously, how to estimate all of these by maximum likelihood; the MLE for the βs is the same as their least-

[1]Presuming, of course, that the model's assumptions, when thoroughly checked, do in fact hold good.

squares estimates. These are

$$\begin{aligned}
\hat{\beta}_1 &= \frac{c_{XY}}{s^2_X} = \sum_{i=1}^{n} \frac{x_i - \overline{x}}{n s^2_X} y_i && (7.1)\\
\hat{\beta}_0 &= \overline{y} - \hat{\beta}_1 \overline{x} && (7.2)\\
\hat{\sigma}^2 &= \frac{1}{n}\sum_{i=1}^{n} (y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i))^2 && (7.3)
\end{aligned}$$

We have also seen how to re-write the first two of these as a deterministic part plus a weighted sum of the noise terms ϵ:

$$\begin{aligned}
\hat{\beta}_1 &= \beta_1 + \sum_{i=1}^{n} \frac{x_i - \overline{x}}{n s^2_X} \epsilon_i && (7.4)\\
\hat{\beta}_0 &= \beta_0 + \frac{1}{n}\sum_{i=1}^{n} \left(1 - \overline{x}\frac{x_i - \overline{x}}{s^2_X}\right)\epsilon_i && (7.5)
\end{aligned}$$

Finally, we have our modeling assumption that the ϵ_i are independent Gaussians, $\epsilon_i \sim N(0, \sigma^2)$.

7.1.1 Reminders of Basic Properties of Gaussian Distributions

Suppose $U \sim N(\mu, \sigma^2)$. By the basic algebra of expectations and variances, $\mathbb{E}[a + bU] = a + b\mu$, while $\mathrm{Var}[a + bU] = b^2\sigma^2$. This would be true of any random variable; a special property of Gaussians[2] is that $a + bU \sim N(a + b\mu, b^2\sigma^2)$.

Suppose $U_1, U_2, \ldots U_n$ are *independent* Gaussians, with means μ_i and variances σ^2_i. Then

$$\sum_{i=1}^{n} U_i \sim N(\sum_i \mu_i, \sum_i \sigma^2_i)$$

That the expected values add up for a sum is true of all random variables; that the variances add up is true for all uncorrelated random variables. That the sum follows the same type of distribution as the summands is a special property of Gaussians[3].

7.1.2 Sampling Distribution of $\hat{\beta}_1$

Since we're assuming Gaussian noise, the ϵ_i are independent Gaussians, $\epsilon_i \sim N(0, \sigma^2)$. Hence (using the first basic property of Gaussians)

$$\frac{x_i - \overline{x}}{n s^2_X}\epsilon_i \sim N(0, \left(\frac{x_i - \overline{x}}{n s^2_X}\right)^2 \sigma^2)$$

[2]There some other families of distributions which have this property; they're called "location-scale" families.

[3]There are some other families of distributions which have this property; they're called "stable" families.

```
# Simulate a Gaussian-noise simple linear regression model Inputs: x
# sequence; intercept; slope; noise variance; switch for whether to return
# the simulated values, or run a regression and return the coefficients
# Output: data frame or coefficient vector
sim.gnslrm <- function(x, intercept, slope, sigma.sq, coefficients = TRUE) {
    n <- length(x)
    y <- intercept + slope * x + rnorm(n, mean = 0, sd = sqrt(sigma.sq))
    if (coefficients) {
        return(coefficients(lm(y ~ x)))
    } else {
        return(data.frame(x = x, y = y))
    }
}

# Fix an arbitrary vector of x's
x <- seq(from = -5, to = 5, length.out = 42)
```

FIGURE 7.1: *Code setting up a simulation of a Gaussian-noise simple linear regression model, along a fixed vector of x_i values.*

Thus, using the second basic property of Gaussians,

$$\sum_{i=1}^{n} \frac{x_i - \overline{x}}{n s_X^2} \epsilon_i \sim N(0, \sigma^2 \sum_{i=1}^{n} \left(\frac{x_i - \overline{x}}{n s_X^2}\right)^2) \tag{7.6}$$

$$= N(0, \frac{\sigma^2}{n s_X^2}) \tag{7.7}$$

Using the first property of Gaussians again,

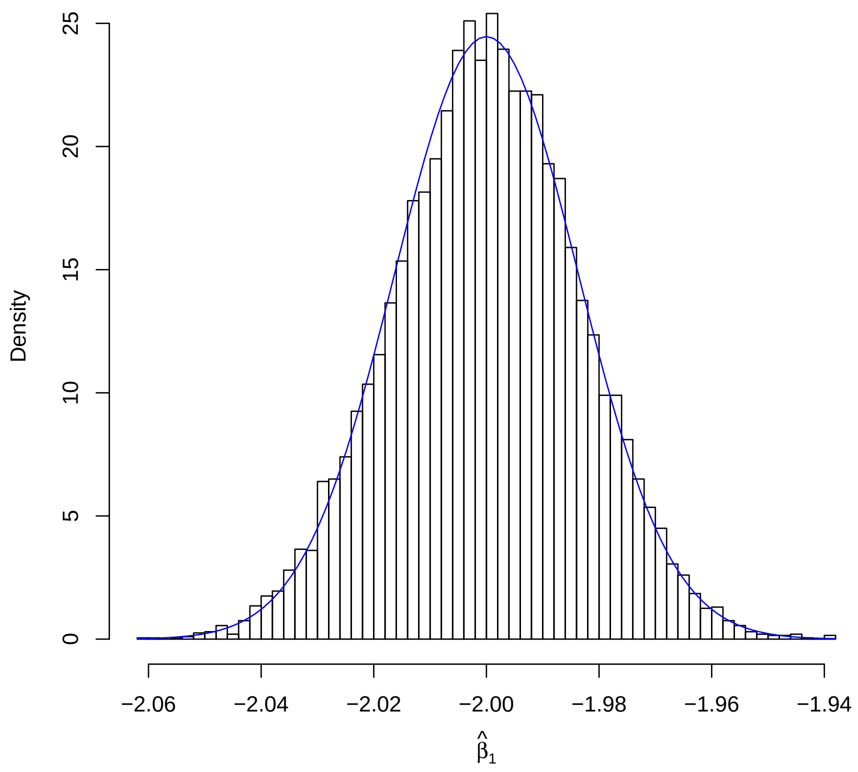
$$\hat{\beta}_1 \sim N(\beta_1, \frac{\sigma^2}{n s_X^2}) \tag{7.8}$$

This is the distribution of estimates we'd see if we repeated the experiment (survey, observation, etc.) many times, and collected the results. Every particular run of the experiment would give a slightly different $\hat{\beta}_1$, but they'd average out to β_1, the average squared difference from β_1 would be $\sigma^2/n s_X^2$, and a histogram of them would follow the Gaussian probability density function (Figure 7.2).

It is a bit hard to use Eq. 7.8, because it involves two of the unknown parameters. We can manipulate it a bit to remove one of the parameters from the probability distribution,

$$\hat{\beta}_1 - \beta_1 \sim N(0, \frac{\sigma^2}{n s_X^2})$$

but that still has σ^2 on the right hand side, so we can't actually calculate anything. We



```
# Run the simulation 10,000 times and collect all the coefficients What
# intercept, slope and noise variance does this impose?
many.coefs <- replicate(10000, sim.gnslrm(x = x, 5, -2, 0.1, coefficients = TRUE))
# Histogram of the slope estimates
hist(many.coefs[2, ], breaks = 50, freq = FALSE, xlab = expression(hat(beta)[1]),
    main = "")
# Theoretical Gaussian sampling distribution
theoretical.se <- sqrt(0.1/(length(x) * var(x)))
curve(dnorm(x, mean = -2, sd = theoretical.se), add = TRUE, col = "blue")
```

FIGURE 7.2: *Simulating 10,000 runs of a Gaussian-noise simple linear regression model, calculating $\hat{\beta}_1$ each time, and comparing the histogram of estimates to the theoretical Gaussian distribution (Eq. 7.8, in blue).*

could write

$$\frac{\hat{\beta}_1 - \beta_1}{\sigma^2 / \sqrt{n s_X^2}} \sim N(0,1)$$

but now we've got two unknown parameters on the left-hand side, which is also awkward.

7.1.3 Sampling Distribution of $\hat{\beta}_0$

Starting from Eq. 7.5 rather than Eq. 7.4, an argument exactly parallel to the one we just went through gives

$$\hat{\beta}_0 \sim N(\beta_0, \frac{\sigma^2}{n}\left(1 + \frac{\overline{x}^2}{s_X^2}\right))$$

It follows, again by parallel reasoning, that

$$\frac{\hat{\beta}_0 - \beta_0}{\sqrt{\frac{\sigma^2}{n}\left(1 + \frac{\overline{x}^2}{s_X^2}\right)}} \sim N(0,1)$$

The right-hand side of this equation is admirably simple and easy for us to calculate, but the left-hand side unfortunately involves two unknown parameters, and that complicates any attempt to use it.

7.1.4 Sampling Distribution of $\hat{\sigma}^2$

It is mildly challenging, but certainly not too hard, to show that

$$\mathbb{E}\left[\hat{\sigma}^2\right] = \frac{n-2}{n}\sigma^2$$

As I have said before, this will be a problem on a future assignment, so I will not give a proof, but I will note that the way to proceed is to write

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n} e_i^2 \ ;$$

then to write each residual e_i as a weighted sum of the noise terms ϵ; to use $\mathbb{E}\left[e_i^2\right] = \mathrm{Var}\left[e_i\right] + (\mathbb{E}\left[e_i\right])^2$; and finally to sum up over i.

Notice that this implies that $\mathbb{E}\left[\hat{\sigma}^2\right] = 0$ when $n = 2$. This is because any two points in the plane define a (unique) line, so if we have only two data points, least squares will just run a line through them exactly, and have an in-sample MSE of 0. In general, we get the factor of $n-2$ from the fact that we are estimating two parameters.

We can however be much more specific. When $\epsilon_i \sim N(0, \sigma^2)$, it can be shown that

$$\frac{n\hat{\sigma}^2}{\sigma^2} \sim \chi^2_{n-2}$$

Notice, by the way, that this equation involves no unknown parameters on the right-hand side, and only one on the left-hand side. It lets us calculate the probability that $\hat{\sigma}^2$ is within any given *factor* of σ^2. If, for instance, we wanted to know the probability that $\hat{\sigma}^2 \geq 7\sigma^2$, this will let us find it.

I will offer only a hand-waving explanation; I am afraid I am not aware of any truly elementary mathematical explanation — every one I know of either uses probability facts which are about as obscure as the result to be shown, or linear-algebraic facts about the properties of idempotent matrices[4], and we've not seen, *yet*, how to write linear regression in matrix form. I do however want to re-assure you that there are actual proofs, and I promise to include one in these notes once we've seen how to connect what we're doing to matrices and linear algebra.

I am afraid I do not have even a hand-waving explanation of a second important property of $\hat{\sigma}^2$: it is statistically independent of $\hat{\beta}_0$ and $\hat{\beta}_1$. This is *not* obvious — after all, all three of these estimators are functions of the same noise variables ϵ — but it *is* true, and, again, I promise to provide a genuine proof in these notes once we've gone over the necessary math.

7.1.4.1 The Hand-Waving Explanation for $n-2$

Let's think for a moment about a related (but strictly different!) quantity from $\hat{\sigma}^2$, namely

$$\frac{1}{n}\sum_{i=1}^{n}\epsilon_i^2$$

This is a weighted sum of independent, mean-zero squared Gaussians, which is where the connection to χ^2 distributions comes in.

Some reminders about χ^2 If $Z \sim N(0,1)$, then $Z^2 \sim \chi^2_1$ *by definition* (of the χ^2_1 distribution). From this, it follows that $\mathbb{E}\left[\chi^2_1\right] = 1$, $\mathrm{Var}\left[\chi^2_1\right] = \mathbb{E}\left[Z^4\right] - (\mathbb{E}\left[Z^2\right])^2 = 2$. If $Z_1, Z_2, \ldots Z_d \sim N(0,1)$ and are independent, then the χ^2_d distribution is *defined* to be the distribution of $\sum_{i=1}^{d} Z_i^2$. By simple algebra, it follows that $\mathbb{E}\left[\chi^2_d\right] = d$ while $\mathrm{Var}\left[\chi^2_d\right] = 2d$.

Back to the sum of squared noise terms ϵ_i isn't a standard Gaussian, but ϵ_i/σ is, so

$$\frac{\sum_{i=1}^{n}\epsilon_i^2}{\sigma^2} = \sum_{i=1}^{n}\left(\frac{\epsilon_i}{\sigma}\right)^2 \sim \chi^2_n$$

The numerator here is *like* $n\hat{\sigma}^2 = \sum_i e_i^2$, but of course the residuals e_i are not the same as the noise terms ϵ_i.

The reason we end up with a χ^2_{n-2} distribution, rather than a χ^2_n distribution, is that we're estimating two parameters from the data removes two degrees of freedom, so two of the ϵ_i end up making no real contribution to the sum of squared errors. (Again, if $n = 2$, we'd be able to fit the two data points *exactly* with the least squares

[4]Where $M^2 = M$.

line.) If we had estimated more or fewer parameters, we would have had to adjust the number of degrees of freedom accordingly.

(There is also a geometric interpretation: the sum of squared errors, $\sum_{i=1}^{n} e_i^2$, is the squared length of the n-dimensional vector of residuals, $(e_1, e_2, \ldots e_n)$. But the residuals must obey the two equations $\sum_i e_i = 0$, $\sum_i x_i e_i = 0$, so the residual vector actually is confined to an $(n-2)$-dimensional linear subspace. Thus we only end up adding up $(n-2)$ *independent* contributions to its length. If we estimated more parameters, we'd have more estimating equations, and so more constraints on the vector of residuals.)

7.1.5 Standard Errors of $\hat{\beta}_0$ and $\hat{\beta}_1$

The **standard error** of an estimator is its standard deviation[5]. We've just seen that the true standard errors of $\hat{\beta}_0$ and $\hat{\beta}_1$ are, respectively,

$$\mathrm{se}\left[\hat{\beta}_1\right] = \frac{\sigma}{s_x\sqrt{n}} \tag{7.9}$$

$$\mathrm{se}\left[\hat{\beta}_0\right] = \frac{\sigma}{\sqrt{n}s_X}\sqrt{s_X^2 + \overline{x}^2} \tag{7.10}$$

Unfortunately, these standard errors involve the unknown parameter σ^2 (or its square root σ, equally unknown to us).

We can, however, *estimate* the standard errors. The maximum-likelihood estimates just substitute $\hat{\sigma}$ for σ:

$$\widehat{\mathrm{se}}\left[\hat{\beta}_1\right] = \frac{\hat{\sigma}}{s_x\sqrt{n}} \tag{7.11}$$

$$\widehat{\mathrm{se}}\left[\hat{\beta}_0\right] = \frac{\hat{\sigma}}{s_X\sqrt{n}}\sqrt{s_X^2 + \overline{x}^2} \tag{7.12}$$

For later theoretical purposes, however, things will work out slightly nicer if we use the de-biased version, $\frac{n}{n-2}\hat{\sigma}^2$:

$$\widehat{\mathrm{se}}\left[\hat{\beta}_1\right] = \frac{\hat{\sigma}}{s_x\sqrt{n-2}} \tag{7.13}$$

$$\widehat{\mathrm{se}}\left[\hat{\beta}_0\right] = \frac{\hat{\sigma}}{s_x\sqrt{n-2}}\sqrt{s_X^2 + \overline{x}^2} \tag{7.14}$$

These standard errors — approximate or estimated though they be — are one important way of quantifying how much uncertainty there is around our point estimates. However, we can't use them, *alone* to say anything terribly precise[6] about, say, the probability that β_1 is in the interval $[\hat{\beta}_1 - \widehat{\mathrm{se}}\left[\hat{\beta}_1\right], \hat{\beta}_1 - \widehat{\mathrm{se}}\left[\hat{\beta}_1\right]]$, which is

[5]We don't just call it the standard deviation because we want to emphasize that it is, in fact, telling us about the random errors our estimator makes.

[6]Exercise to think through: Could you use Chebyshev's inequality (the extra credit problem from Homework 1) here?

the sort of thing we'd want to be able to give guarantees about the reliability of our estimates.

7.2 Sampling distribution of $(\hat{\beta}-\beta)/\widehat{\mathrm{se}}\left[\hat{\beta}\right]$

It should take only a little work with the properties of the Gaussian distribution to convince yourself that

$$\frac{\hat{\beta}_1-\beta_1}{\mathrm{se}\left[\hat{\beta}_1\right]} \sim N(0,1)$$

the standard Gaussian distribution. If the Oracle told us σ^2, we'd know $\mathrm{se}\left[\hat{\beta}_1\right]$, and so we could assert that (for example)

$$\begin{aligned}
& \mathbb{P}\left(\beta_1-1.96\mathrm{se}\left[\hat{\beta}_1\right] \leq \hat{\beta}_1 \leq \beta_1+1.96\mathrm{se}\left[\hat{\beta}_1\right]\right) && (7.15)\\
= \ & \mathbb{P}\left(-1.96\mathrm{se}\left[\hat{\beta}_1\right] \leq \hat{\beta}_1-\beta_1 \leq 1.96\mathrm{se}\left[\hat{\beta}_1\right]\right) && (7.16)\\
= \ & \mathbb{P}\left(-1.96 \leq \frac{\hat{\beta}_1-\beta_1}{\mathrm{se}\left[\hat{\beta}_1\right]} \leq 1.96\right) && (7.17)\\
= \ & \Phi(1.96)-\Phi(-1.96) = 0.95 && (7.18)
\end{aligned}$$

where Φ is the cumulative distribution function of the $N(0,1)$ distribution.

Since the oracles have fallen silent, we can't use this approach. What we *can* do is use the following fact[7]:

Proposition 1 *If $Z \sim N(0,1)$, $S^2 \sim \chi^2_d$, and Z and S^2 are independent, then*

$$\frac{Z}{\sqrt{S^2/d}} \sim t_d$$

(I call this a proposition, but it's almost a definition of what we mean by a t distribution with d degrees of freedom. Of course, if we take this as the definition, the proposition that this distribution has a probability density $\propto (1+x^2/d)^{-(d+1)/2}$ would become yet another proposition to be demonstrated.)

Let's try to manipulate $(\hat{\beta}_1-\beta_1)/\widehat{\mathrm{se}}\left[\hat{\beta}_1\right]$ into this form.

[7]When I messed up the derivation in class today, I left out dividing by d in the denominator. As I mentioned at the end of that debacle, this was stupid. As $d \to \infty$, t_d converges on the standard Gaussian distribution $N(0,1)$. (Notice that $\mathbb{E}\left[d^{-1}\chi^2_d\right]=1$, while $\mathrm{Var}\left[d^{-1}\chi^2_d\right]=2/d$, so $d^{-1}\chi^2_d \to 1$.) Without the normalizing factor of d inside the square root, however, looking just at Z/S, we've got a random variable whose distribution doesn't change with d being divided by something whose magnitude *grows* with d, so $Z/S \to 0$ as $d \to \infty$, not $\to N(0,1)$. I apologize again for my error.

$$\begin{aligned}
\frac{\hat{\beta}_1-\beta_1}{\widehat{\mathrm{se}}\left[\hat{\beta}_1\right]} &= \frac{\hat{\beta}_1-\beta_1}{\sigma}\frac{\sigma}{\widehat{\mathrm{se}}\left[\hat{\beta}_1\right]} && (7.19)\\
&= \frac{\frac{\hat{\beta}_1-\beta_1}{\sigma}}{\frac{\widehat{\mathrm{se}}\left[\hat{\beta}_1\right]}{\sigma}} && (7.20)\\
&= \frac{N(0,1/ns_X^2)}{\frac{\hat{\sigma}}{s_x\sigma\sqrt{n-2}}} && (7.21)\\
&= \frac{s_X N(0,1/ns_X^2)}{\frac{\hat{\sigma}}{\sigma\sqrt{n-2}}} && (7.22)\\
&= \frac{N(0,1/n)}{\frac{\hat{\sigma}}{\sigma\sqrt{n-2}}} && (7.23)\\
&= \frac{\sqrt{n}N(0,1/n)}{\frac{\sqrt{n}\hat{\sigma}}{\sigma\sqrt{n-2}}} && (7.24)\\
&= \frac{N(0,1)}{\sqrt{\frac{n\hat{\sigma}^2}{\sigma^2}\frac{1}{n-2}}} && (7.25)\\
&= \frac{N(0,1)}{\sqrt{\chi^2_{n-2}/(n-2)}} && (7.26)\\
&= t_{n-2} && (7.27)
\end{aligned}$$

where in the last step I've used the proposition I stated (without proof) above.

To sum up:

Proposition 2 *Using the* $\widehat{\mathrm{se}}\left[\hat{\beta}_1\right]$ *of Eq. 7.13,*

$$\frac{\hat{\beta}_1-\beta_1}{\widehat{\mathrm{se}}\left[\hat{\beta}_1\right]} \sim t_{n-2} \tag{7.28}$$

Notice that we can compute $\widehat{\mathrm{se}}\left[\hat{\beta}_1\right]$ without knowing any of the true parameters — it's a pure statistic, just a function of the data. This is a key to actually using the proposition for anything useful.

By exactly parallel reasoning, we may also demonstrate that

$$\frac{\hat{\beta}_0-\beta_0}{\widehat{\mathrm{se}}\left[\hat{\beta}_0\right]} \sim t_{n-2}$$

7.3 Sampling Intervals for $\hat{\beta}$; hypothesis tests for $\hat{\beta}$

Let's trace through one of the consequences of Eq. 7.28. For any $k > 0$,

$$\mathbb{P}\left(\beta_1 - k\widehat{\mathrm{se}}\left[\hat{\beta}_1\right] \leq \hat{\beta}_1 \leq \beta_1 + k\widehat{\mathrm{se}}\left[\hat{\beta}_1\right]\right) \tag{7.29}$$

$$= \mathbb{P}\left(k\widehat{\mathrm{se}}\left[\hat{\beta}_1\right] \leq \hat{\beta}_1 - \beta_1 \leq k\widehat{\mathrm{se}}\left[\hat{\beta}_1\right]\right) \tag{7.30}$$

$$= \mathbb{P}\left(k \leq \frac{\hat{\beta}_1 - \beta_1}{\widehat{\mathrm{se}}\left[\hat{\beta}_1\right]} \leq k\right) \tag{7.31}$$

$$= \int_{-k}^{k} t_{n-2}(u)du \tag{7.32}$$

where by a slight abuse of notation I am writing $t_{n-2}(u)$ for the probability density of the t distribution with $n-2$ degrees of freedom, evaluated at the point u.

It should be evident that if you pick any α between 0 and 1, I can find a $k(n,\alpha)$ such that

$$\int_{-k(n,\alpha)}^{k(n,\alpha)} t_{n-2}(u)du = 1-\alpha$$

I therefore define the (symmetric) $1-\alpha$ **sampling interval** for $\hat{\beta}_1$, when the true slope is β_1, as

$$[\beta_1 - k(n,\alpha)\widehat{\mathrm{se}}\left[\hat{\beta}_1\right], \beta_1 + k(n,\alpha)\widehat{\mathrm{se}}\left[\hat{\beta}_1\right]] \tag{7.33}$$

If the true slope is β_1, then $\hat{\beta}_1$ will be within this sampling interval with probability $1-\alpha$. This leads directly to a test of the null hypothesis that the slope $\beta_1 = \beta_1^*$: reject the null if $\hat{\beta}_1$ is outside the sampling interval for β_1^*, and retain the null when $\hat{\beta}_1$ is inside that sampling interval. This test is called the **Wald test**, after the great statistician Abraham Wald[8].

By construction, the Wald test's probability of rejection under the null hypothesis — the **size**, or **type I error rate**, or **false alarm rate** of the test — is exactly α. Of course, the other important property of a hypothesis test is its **power** — the probability of rejecting the null when it is false. From Eqn. 7.28, it should be clear that if the true $\beta_1 \neq \beta_1^*$, the probability that $\hat{\beta}_1$ is inside the sampling interval for β_1^* is $< 1-\alpha$, with the difference growing as $|\beta_1 - \beta_1^*|$ grows. An exact calculation could be done (it'd involve what's called the "non-central t distribution"), but is not especially informative. The point is that the power is always $> \alpha$, and grows with the departure from the null hypothesis.

If you were an economist, psychologist, or something of their ilk, you have a powerful drive — almost a spinal reflex not involving the higher brain regions — to

[8]As is common with eponyms in the sciences, Wald was not, in fact, the first person to use the test, but he made one of the most important early studies of its properties, and he was already famous for other reasons.

test whether $\beta_1 = 0$. Under the Wald test, you would reject that point null hypothesis when $|\hat{\beta}_1|$ exceeds a certain number of standard deviations. As an intelligent statistician in control of your own actions, you would read the section on "statistical significance" below, before doing any such thing.

All of the above applies, *mutatis mutandis*, to $\frac{\hat{\beta}_0-\beta_0}{\widehat{\mathrm{se}}\left[\hat{\beta}_0\right]}$.

7.4 Building Confidence Intervals from Sampling Intervals

Once we know how to calculate sampling intervals, we can plot the sampling interval for every possible value of β_1 (Figure 7.3). They're the region marked off by two parallel lines, one $k(n,\alpha)\widehat{\mathrm{se}}\left[\hat{\beta}_1\right]$ above the main diagonal and one equally far below the main diagonal.

The sampling intervals (as in Figure 7.3) are theoretical constructs — mathematical consequences of the assumptions in the the probability model that (we hope) describes the world. After we gather data, we can actually calculate $\hat{\beta}_1$. This is a random quantity, but it will have some particular value on any data set. We can mark this realized value, and draw a horizontal line across the graph at that height (Figure 7.4).

The $\hat{\beta}_1$ we observed is within the sampling interval for some (possible or hypothetical) values of β_1, and outside the sampling interval for others. We define the **confidence set**, with **confidence level** $1-\alpha$, as

$$\left\{\beta_1 \,:\, \hat{\beta}_1 \in [\beta_1 - k(n,\alpha)\widehat{\mathrm{se}}\left[\hat{\beta}_1\right], \beta_1 + k(n,\alpha)\widehat{\mathrm{se}}\left[\hat{\beta}_1\right]]\right\} \tag{7.34}$$

This is precisely the set of β_1 which we retain when we run the Wald test with size α. In other words: we test every possible β_1; if we'd reject that null hypothesis, that value of β_1 gets removed from the hypothesis test; if we'd retain that null, β_1 stays in the confidence set[9]. Figure 7.5 illustrate a confidence set, and shows (unsurprisingly) that in this case the confidence set is indeed a confidence *interval*. Indeed, a little manipulation of Eq. 7.34 gives us an explicit formula for the confidence set, which is an interval:

Confidence set = Test all the hypotheses!

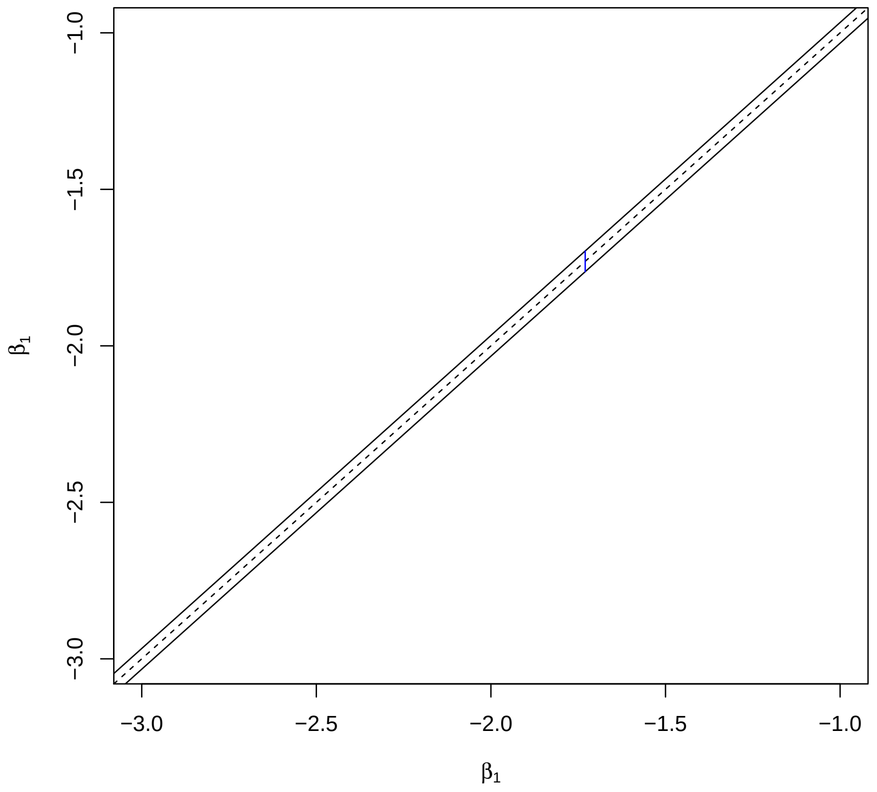
$$[\hat{\beta}_1 - k(n,\alpha)\widehat{\mathrm{se}}\left[\hat{\beta}_1\right], \hat{\beta}_1 + k(n,\alpha)\widehat{\mathrm{se}}\left[\hat{\beta}_1\right]$$

The correct interpretation of a confidence set is that it offers us a dilemma. One of two[10] things must be true:

1. The true β_1 is inside the confidence set.

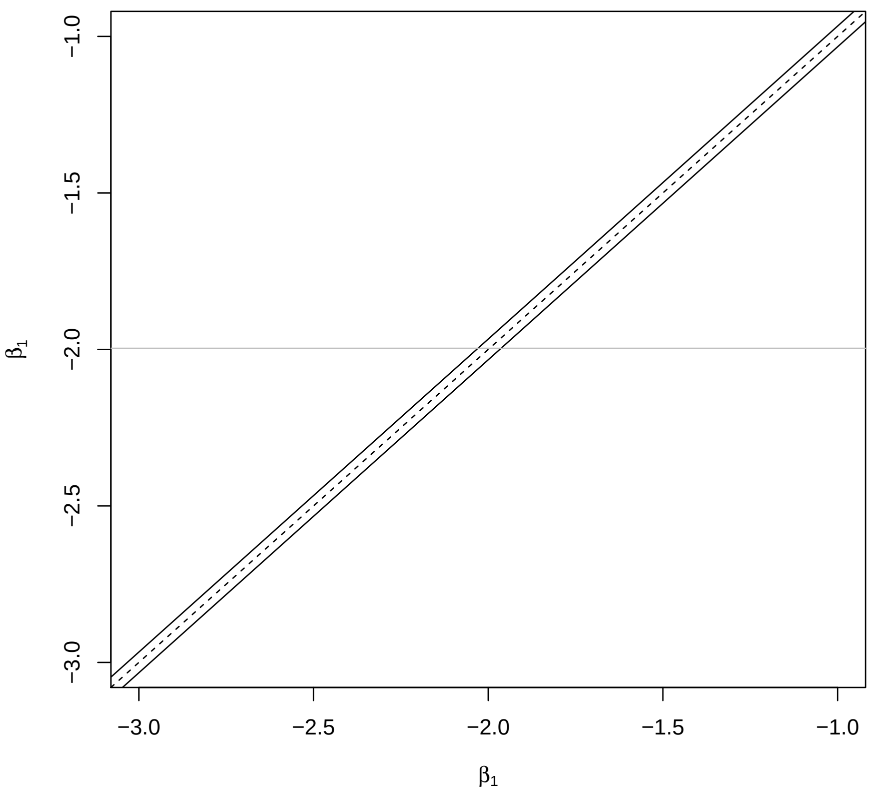
[9]Cf. the famous Sherlock Holmes line "When you have eliminated the impossible, whatever remains, however improbable, must be the truth." In forming the confidence set, we are eliminating the merely *unlikely*, rather than the absolutely impossible. This is because, not living in a detective story, we get only noisy and imperfect evidence.

[10]Strictly speaking, there is a third option: our model could be wrong. Hence the importance of model checking *before* doing within-model inference.



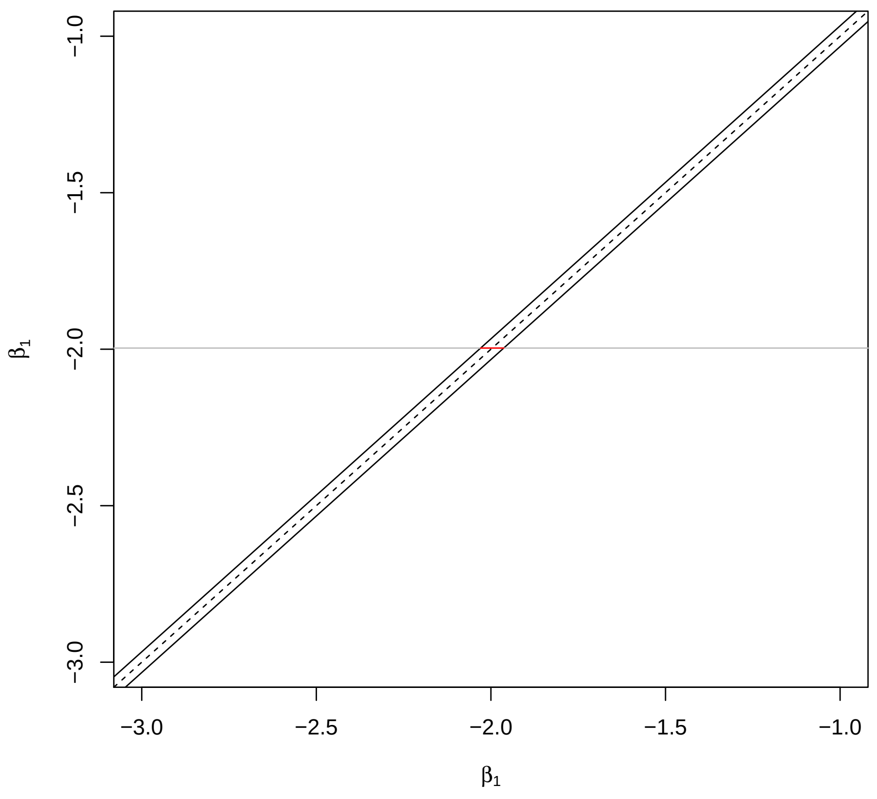
```
lm.sim <- lm(y ~ x, data = sim.gnslrm(x = x, 5, -2, 0.1, coefficients = FALSE))
hat.sigma.sq <- mean(residuals(lm.sim)^2)
se.hat.beta.1 <- sqrt(hat.sigma.sq/(var(x) * (length(x) - 2)))
alpha <- 0.02
k <- qt(1 - alpha/2, df = length(x) - 2)
plot(0, xlim = c(-3, -1), ylim = c(-3, -1), type = "n", xlab = expression(beta[1]),
    ylab = expression(hat(beta)[1]), main = "")
abline(a = k * se.hat.beta.1, b = 1)
abline(a = -k * se.hat.beta.1, b = 1)
abline(a = 0, b = 1, lty = "dashed")
beta.1.star <- -1.73
segments(x0 = beta.1.star, y0 = k * se.hat.beta.1 + beta.1.star, x1 = beta.1.star,
    y1 = -k * se.hat.beta.1 + beta.1.star, col = "blue")
```

FIGURE 7.3: *Sampling intervals for $\hat{\beta}_1$ as a function of β_1. For compatibility with the earlier simulation, I have set $n = 42$, $s_X^2 = 9$, and (from one run of the model) $\hat{\sigma}^2 = 0.067$; and, just because $\alpha = 0.05$ is cliched, $\alpha = 0.02$. Equally arbitrarily, the blue vertical line illustrates the sampling interval when $\beta_1 = -1.73$.*



```
plot(0, xlim = c(-3, -1), ylim = c(-3, -1), type = "n", xlab = expression(beta[1]),
    ylab = expression(hat(beta)[1]), main = "")
abline(a = k * se.hat.beta.1, b = 1)
abline(a = -k * se.hat.beta.1, b = 1)
abline(a = 0, b = 1, lty = "dashed")
beta.1.hat <- coefficients(lm.sim)[2]
abline(h = beta.1.hat, col = "grey")
```

FIGURE 7.4: *As in Figure 7.3, but with the addition of a horizontal line marking the observed value of $\hat{\beta}_1$ on a particular realization of the simulation (in grey).*



```
plot(0, xlim = c(-3, -1), ylim = c(-3, -1), type = "n", xlab = expression(beta[1]),
    ylab = expression(hat(beta)[1]), main = "")
abline(a = k * se.hat.beta.1, b = 1)
abline(a = -k * se.hat.beta.1, b = 1)
abline(a = 0, b = 1, lty = "dashed")
beta.1.hat <- coefficients(lm.sim)[2]
abline(h = beta.1.hat, col = "grey")
segments(x0 = beta.1.hat - k * se.hat.beta.1, y0 = beta.1.hat, x1 = beta.1.hat +
    k * se.hat.beta.1, y1 = beta.1.hat, col = "red")
```

FIGURE 7.5: *As in Figure 7.4, but with the* **confidence set** *marked in red. This is the collection of all* β_1 *where* $\hat{\beta}_1$ *falls within the* $1-\alpha$ *sampling interval.*

2. $\hat{\beta}_1$ is outside the sampling interval of the true β_1.

We know that the second option has probability at most α, no matter what the true β_1 is, so we may rephrase the dilemma. Either

1. The true β_1 is inside the confidence set, or

2. We're very unlucky, because something whose probability is $\leq \alpha$ happened.

Since, most of the time, we're not very unlucky, the confidence set is, in fact, a reliable way of giving a margin of error for the true parameter β_1.

Width of the confidence interval Notice that the width of the confidence interval is $2k(n,\alpha)\widehat{\text{se}}\left[\hat{\beta}_1\right]$. This tells us what controls the width of the confidence interval:

1. As α shrinks, the interval widens. (High confidence comes at the price of big margins of error.)

2. As n grows, the interval shrinks. (Large samples mean precise estimates.)

3. As σ^2 increases, the interval widens. (The more noise there is around the regression line, the less precisely we can measure the line.)

4. As s_X^2 grows, the interval shrinks. (Widely-spread measurements give us a precise estimate of the slope.)

What about β_0? By exactly parallel reasoning, a $1-\alpha$ confidence interval for β_0 is $[\hat{\beta}_0 - k(n,\alpha)\widehat{\text{se}}\left[\hat{\beta}_0\right], \hat{\beta}_0 + k(n,\alpha)\widehat{\text{se}}\left[\hat{\beta}_0\right]]$.

What about σ^2? See Exercise 1.

What α should we use? It's become conventional to set $\alpha = 0.05$. To be honest, this owes more to the fact that the resulting k tends to 1.96 as $n \to \infty$, and $1.96 \approx 2$, and most psychologists and economists could multiply by 2, even in 1950, than to any genuine principle of statistics or scientific method. A 5% error rate corresponds to messing up about one working day in every month, which you might well find high. On the other hand, there is nothing which stops you from increasing α. It's often illuminating to plot a series of confidence sets, at different values of α.

What about power? The **coverage** of a confidence set is the probability that it includes the true parameter value. This is not, however, the only virtue we want in a confidence set; if it was, we could just say "Every possible parameter is in the set", and have 100% coverage no matter what. We would also like the *wrong* values of the parameter to have a high probability of *not* being in the set. Just as the coverage is controlled by the size / false-alarm probability / type-I error rate α of the hypothesis test, the probability of excluding the wrong parameters is controlled by the power / miss probability / type-II error rate. Test with higher power exclude (correctly) more parameter values, and give smaller confidence sets.

7.4.1 Confidence Sets and Hypothesis Tests

I have derived confidence sets for β by inverting a specific hypothesis test, the Wald test. There is a more general relationship between confidence sets and hypothesis tests.

1. Inverting any hypothesis test gives us a confidence set.
2. If we have a way of constructing a $1-\alpha$ confidence set, we can use it to test the hypothesis that $\beta = \beta^*$: reject when β^* is outside the confidence set, retain the null when β^* is inside the set.

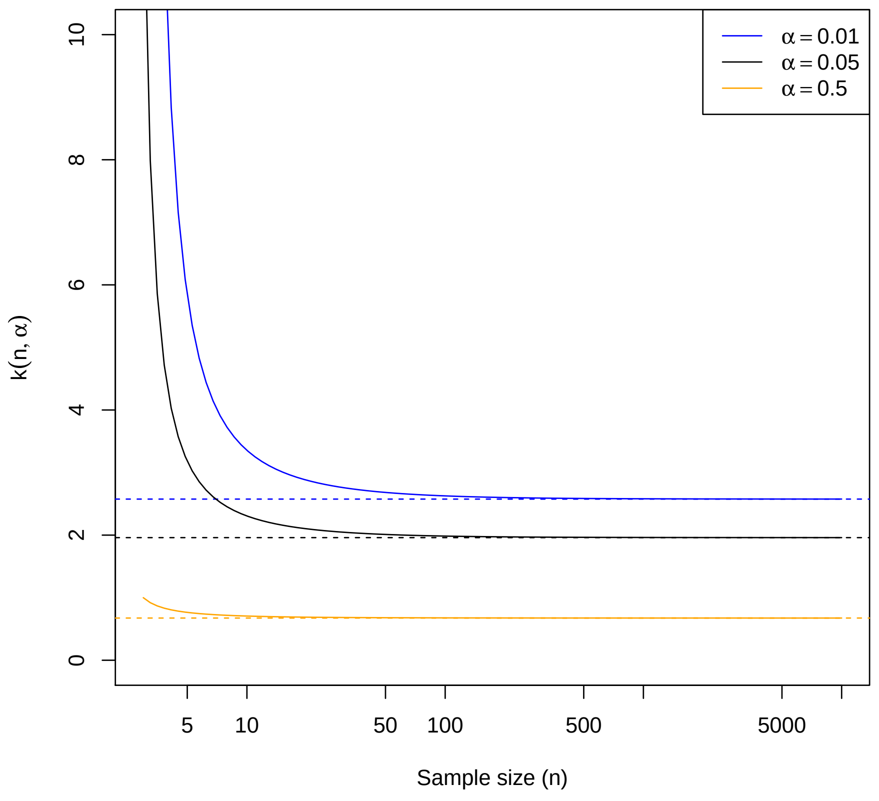
I will leave it as a pair of exercises (2 and 3) to that inverting a test of size α gives a $1-\alpha$ confidence set, and that inverting a $1-\alpha$ confidence set gives a test of size α.

7.4.2 Large-n Asymptotics

As $n \rightarrow \infty$, $\hat{\sigma}^2 \rightarrow \sigma^2$. It follows (by continuity) that $\widehat{\text{se}}\left[\hat{\beta}\right] \rightarrow \text{se}\left[\hat{\beta}\right]$. Hence,

$$\frac{\hat{\beta}-\beta}{\widehat{\text{se}}\left[\hat{\beta}\right]} \rightarrow N(0,1)$$

which considerably simplifies the sampling intervals and confidence sets; as n grows, we can forget about the t distribution and just use the standard Gaussian distribution. Figure 7.6 plots the convergence of $k(n,\alpha)$ towards the $k(\infty,\alpha)$ we'd get from the Gaussian approximation. As you can see from the figure, by the time $n = 100$ —a quite small data set by modern standards — the difference between the t distribution and the standard-Gaussian is pretty trivial.



```
curve(qt(0.995, df = x - 2), from = 3, to = 10000, log = "x", ylim = c(0, 10),
    xlab = "Sample size (n)", ylab = expression(k(n, alpha)), col = "blue")
abline(h = qnorm(0.995), lty = "dashed", col = "blue")
curve(qt(0.975, df = x - 2), add = TRUE)
abline(h = qnorm(0.975), lty = "dashed")
curve(qt(0.75, df = x - 2), add = TRUE, col = "orange")
abline(h = qnorm(0.75), lty = "dashed", col = "orange")
legend("topright", legend = c(expression(alpha == 0.01), expression(alpha ==
    0.05), expression(alpha == 0.5)), col = c("blue", "black", "orange"), lty = "solid")
```

FIGURE 7.6: *Convergence of $k(n, \alpha)$ as $n \to \infty$, illustrated for $\alpha = 0.01$, $\alpha = 0.05$ and $\alpha = 0.5$. (Why do I plot the 97.5th percentile when I'm interested in $\alpha = 0.05$?)*

7.5 Statistical Significance: Uses and Abuses

7.5.1 p-Values

The test statistic for the Wald test,

$$T = \frac{\hat{\beta}_1 - \beta_1^*}{\widehat{\mathrm{se}}\left[\hat{\beta}_1\right]}$$

has the nice, intuitive property that it ought to be close to zero when the null hypothesis $\beta_1 = \beta_1^*$ is true, and take large values (either positive or negative) when the null hypothesis is false. When a test statistic works like this, it makes sense to summarize just how bad the data looks for the null hypothesis in a **p-value**: when our observed value of the test statistic is T_{obs}, the p-value is

$$P = \mathbb{P}(|T| \geq |T_{obs}|)$$

calculating the probability under the null hypothesis. (I write a capital P here as a reminder that this is a random quantity, though it's conventional to write the phrase "p-value" with a lower-case p.) This is the probability, under the null, of getting results which are at least as extreme as what we saw. It should be easy to convince yourself that rejecting the null in a level-α test is the same as getting a p-value $< \alpha$.

It is not too hard (Exercise 4) to show that P has a uniform distribution over $[0, 1]$ under the null hypothesis.

7.5.2 p-Values and Confidence Sets

When our test lets us calculate a p-value, we can form a $1-\alpha$ confidence set by taking all the β's where the p-value is $\geq \alpha$. Conversely, if we have some way of making confidence sets already, we can get a p-value for the hypothesis $\beta = \beta^*$; it's the largest α such that β^* is in the $1-\alpha$ confidence set.

7.5.3 Statistical Significance

If we test the hypothesis that $\beta_1 = \beta_1^*$ and reject it, we say that the difference between β_1 and β_1^* is **statistically significant**. Since, as I mentioned, many professions have an overwhelming urge to test the hypothesis $\beta_1 = 0$, it's common to hear people say that "β_1 is statistically significant" when they mean "β_1 is difference from 0 is statistically significant".

This is harmless enough, as long as we keep firmly in mind that "significant" is used here as a technical term, with a special meaning, and is *not* the same as "important", "relevant", etc. When we reject the hypothesis that $\beta_1 = 0$, what we're saying is "It's really implausibly hard to fit this data with a flat line, as opposed to one with a slope". This is informative, if we had serious reasons to think that a flat line was a live option.

It is incredibly common for researchers from other fields, and even some statisticians, to reason as follows: "I tested whether $\beta_1 = 0$ or not, and I retained the null; *therefore* β_1 is *in*significant, and I can ignore it." This is, of course, a complete fallacy.

To see why, it is enough to realize that there are (at least) two reasons why our hypothesis test might retain the null $\beta_1 = 0$:

1. β_1 is, in fact, zero,
2. $\beta_1 \neq 0$, but $\widehat{\mathrm{se}}\left[\hat{\beta}_1\right]$ is so large that we can't tell anything about β_1 with any confidence.

There is a very big difference between data which lets us say "we can be quite confident that the true β_1 is, if not perhaps exactly 0, then very small", and data which only lets us say "we have no earthly idea what β_1 is, and it may as well be zero for all we can tell"[11]. It is good practice to always compute a confidence interval, but it is *especially* important to do so when you retain the null, so you know whether you can say "this parameter is zero to within such-and-such a (small) precision", or whether you have to admit "I couldn't begin to tell you what this parameter is".

Substantive vs. statistical significance Even a huge β_1, which it would be crazy to ignore in any circumstance, can be statistically insignificant, so long as $\widehat{\mathrm{se}}\left[\hat{\beta}_1\right]$ is large enough. Conversely, any β_1 which isn't exactly zero, no matter how close it might be to 0, will become statistically significant at any threshold once $\widehat{\mathrm{se}}\left[\hat{\beta}_1\right]$ is small enough. Since, as $n \rightarrow \infty$,

$$\widehat{\mathrm{se}}\left[\hat{\beta}_1\right] \rightarrow \frac{\sigma}{s_X \sqrt{n}}$$

we can show that $\widehat{\mathrm{se}}\left[\hat{\beta}_1\right] \rightarrow 0$, and $\frac{\hat{\beta}_1}{\widehat{\mathrm{se}}\left[\hat{\beta}_1\right]} \rightarrow \pm\infty$, unless β_1 is exactly 0 (see below).

Statistical significance is a weird mixture of how big the coefficient is, how big a sample we've got, how much noise there is around the regression line, and how spread out the data is along the x axis. This has so little to do with "significance" in ordinary language that it's pretty unfortunate we're stuck with the word; if the Ancestors had decided to say "statistically detectable" or "statistically distinguishable from 0", we might have avoided a lot of confusion.

If *you* confuse substantive and statistical significance in this class, it will go badly for you.

[11]Imagine hearing what sounds like the noise of an animal in the next room. If the room is small, brightly lit, free of obstructions, and you make a thorough search of it with unimpaired vision and concentration, not finding an animal in it is, in fact, good evidence that there was no animal there to be found. If on the other hand the room is dark, large, full of hiding places, and you make a hurried search while distracted, without your contact lenses and after a few too many drinks, you could easily have missed all sorts of things, and your negative report has little weight as evidence. (In this parable, the difference between a large $|\beta_1|$ and a small $|\beta_1|$ is the difference between looking for a Siberian tiger and looking for a little black cat.)

7.5.4 Appropriate Uses of p-Values and Significance Testing

I do not want this section to give the impression that p-values, hypothesis testing, and statistical significance are unimportant or necessarily misguided. They're often used badly, but that's true of every statistical tool from the sample mean on down the line. There are certainly situations where we really do want to know whether we have good evidence against some *exact* statistical hypothesis, and that's just the job these tools do. What are some of these situations?

Model checking Our statistical models often make very strong, claims about the probability distribution of the data, with little wiggle room. The simple linear regression model, for instance, claims that the regression function is *exactly* linear, and that the noise around this line has *exactly* constant variance. If we test these claims and find very small p-values, then we have evidence that there's a detectable, systematic departure from the model assumptions, and we should re-formulate the model.

Actual scientific interest Some scientific theories make very precise predictions about coefficients. According to Newton, the gravitational force between two masses is inversely proportional to the *square* of the distance between them, $\propto r^{-2}$. The prediction is exactly $\propto r^{-2}$, not $\propto r^{-1.99}$ nor $\propto r^{-2.05}$. Measuring that exponent and finding even tiny departures from 2 would be big news, if we had reason to think they were real and not just noise[12]. One of the most successful theories in physics, quantum electrodynamics, makes predictions about some properties of hydrogen atoms with a theoretical precision of one part in a trillion; finding even tiny discrepancies between what the theory predicts and what we estimate would force us to rethink lots of physics[13]. Experiments to detect new particles, like the Higgs boson, essentially boil down to hypothesis testing, looking for deviations from theoretical predictions which should be exactly zero if the particle doesn't exist.

Outside of the natural sciences, however, it is harder to find examples of interesting, exact null hypothesis which are, so to speak, "live options". The best I can come up with are theories of economic growth and business cycles which predict that the share of national income going to labor (as opposed to capital) should be constant over time. Otherwise, in the social sciences, there's usually little theoretical reason to think that certain regression coefficients should be *exactly* zero, or *exactly* one, or anything else.

Neutral models A partial exception is the use of **neutral models**, which comes out of genetics and ecology. The idea here is to check whether some mechanism is at work in a particular situation — say, whether some gene is subject to natural selection. One constructs two models; one incorporates all the mechanisms (which we think are) at work, including the one under investigation, and the other incorporate all the *other* mechanisms, but "neutralizes" the one of interest. (In a genetic example, the neutral

[12]In fact, it *was* big news: Einstein's theory of general relativity.

[13]Feynman (1985) gives a great conceptual overview of quantum electrodynamics. Currently, theory agrees with experiment to the limits of experimental precision, which is only about one part in a billion (`https://en.wikipedia.org/wiki/Precision_tests_of_QED`).

model would probably incorporate the effects of mutation, sexual repdouction, the random sampling of which organisms become the ancestors of the next generation, perhaps migration, etc. The non-neutral model would include all this *plus* the effects of natural selection.) Rejecting the neutral model in favor of the non-neutral one then becomes evidence that the disputed mechanism is needed to explain the data.

In the cases where this strategy has been done well, the neutral model is usually a pretty sophisticated stochastic model, and the "neutralization" is not as simple as just setting some slope to zero. Nonetheless, this is a situation where we do actually learn something about the world by testing a null hypothesis.

7.6 Any Non-Zero Parameter Becomes Significant with Enough Information

(This section is optional, but strongly recommended.)

Let's look more close at what happens to the test statistic when $n \rightarrow \infty$, and so at what happens to the p-value. Throughout, we'll be testing the null hypothesis that $\beta_1 = 0$, since this is what people most often do, but the same reasoning applies to departures from any fixed value of the slope. (Everything carries over with straightforward changes to testing hypotheses about the intercept β_0, too.)

We know that $\hat{\beta}_1 \sim N(\beta_1, \sigma^2/ns_X^2)$. This means[14]

$$\begin{aligned}
\hat{\beta}_1 &\sim \beta_1 + N(0, \sigma^2/ns_X^2) && (7.35)\\
&= \beta_1 + \frac{\sigma}{s_X\sqrt{n}}N(0,1) && (7.36)\\
&= \beta_1 + O(1/\sqrt{n}) && (7.37)
\end{aligned}$$

where $O(f(n))$ is read "order-of $f(n)$", meaning that it's a term whose size grows like $f(n)$ as $n \rightarrow \infty$, and we don't want (or need) to keep track of the details. Similarly, since $n\hat{sigma}^2/\sigma^2 \sim \chi^2_{n-2}$, we have[15]

$$\begin{aligned}
n\hat{\sigma}^2 &\sim \sigma^2\chi^2_{n-2} && (7.38)\\
\hat{\sigma}^2 &\sim \sigma^2\frac{\chi^2_{n-2}}{n} && (7.39)
\end{aligned}$$

Since $\mathbb{E}\left[\chi^2_{n-2}\right] = n-2$ and $\mathrm{Var}\left[\chi^2_{n-2}\right] = 2(n-2)$,

$$\begin{aligned}
\mathbb{E}\left[\frac{\chi^2_{n-2}}{n}\right] &= \frac{n-2}{n} \rightarrow 1 && (7.40)\\
\mathrm{Var}\left[\frac{\chi^2_{n-2}}{n}\right] &= \frac{2(n-2)}{n^2} \rightarrow 0 && (7.41)
\end{aligned}$$

[14]If seeing something like $\frac{\sigma}{s_X\sqrt{n}}N(0,1)$, feel free to introduce random variables $Z_n \sim N(0,1)$ (though not necessarily independent ones), and modify the equations accordingly.

[15]Again, feel free to introduce the random variable Ξ_n, which just so happens to have a χ^2_{n-2} distribution.

with both limits happening as $n \to \infty$. In fact $\mathrm{Var}\left[\frac{\chi^2_{n-2}}{n}\right] = O(1/n)$, so

$$\hat{\sigma}^2 = \sigma^2\left(1 + O(1/\sqrt{n})\right) \tag{7.42}$$

Taking the square root, and using the fact[16] that $(1+x)^a \approx 1 + ax$ when $|x| \ll 1$,

$$\hat{\sigma} = \sigma\left(1 + O(1/\sqrt{n})\right) \tag{7.43}$$

Put this together to look at our test statistic:

$$\begin{aligned}
\frac{\hat{\beta}_1}{\widehat{\mathrm{se}}\left[\hat{\beta}_1\right]} &= \frac{\beta_1 + O(1/\sqrt{n})}{\frac{\sigma\left(1+O(1/\sqrt{n})\right)}{s_X\sqrt{n}}} && (7.44)\\
&= \sqrt{n}\frac{\beta_1 + O(1/\sqrt{n})}{(\sigma/s_X)\left(1+O(1/\sqrt{n})\right)} && (7.45)\\
&= \sqrt{n}\frac{\beta_1}{\sigma/s_X}\left(1+O(1/\sqrt{n})\right) && (7.46)\\
&= \sqrt{n}\frac{\beta_1}{\sigma/s_X} + O(1) && (7.47)
\end{aligned}$$

In words: so long as the true $\beta_1 \neq 0$, the test statistic is going to go off to $\pm\infty$, and the rate at which it escapes towards infinity is going to be proportional to $\sqrt{n}$. When we compare this against the null distribution, which is $N(0,1)$, eventually we'll get arbitrarily small p-values.

We can actually compute what those p-values should be, by two bounds on the standard Gaussian distribution[17]:

$$\left(\frac{1}{x} - \frac{1}{x^3}\right)\frac{1}{\sqrt{2\pi}}e^{-x^2/2} < 1 - \Phi(x) < \frac{1}{x}\frac{1}{\sqrt{2\pi}}e^{-x^2/2} \tag{7.48}$$

Thus

$$\begin{aligned}
P_n &= \mathbb{P}\left(|Z| \geq \left|\frac{\hat{\beta}_1}{\hat{\sigma}/\sqrt{n}s_X}a\right|\right) && (7.49)\\
&= 2\mathbb{P}\left(Z \geq \left|\frac{\hat{\beta}_1}{\hat{\sigma}/\sqrt{n}s_X}\right|\right) && (7.50)\\
&\leq \frac{2}{\sqrt{2\pi}}\frac{e^{-\frac{1}{2}\frac{\hat{\beta}_1^2}{\hat{\sigma}^2/ns_X^2}}}{\left|\frac{\hat{\beta}_1}{\hat{\sigma}/\sqrt{n}s_X}\right|} && (7.51)
\end{aligned}$$

[16]From the binomial theorem, back in high school algebra.

[17]See Feller (1957), Chapter VII, §1, Lemma 2. For a brief proof online, see `http://www.johndcook.com/normalbounds.pdf`.

To clarify the behavior, let's take the logarithm and divide by n:

$$\begin{aligned}
\frac{1}{n}\log P_n &\leq \frac{1}{n}\log\frac{2}{\sqrt{2\pi}} \\
&\quad -\frac{1}{n}\log\left|\frac{\hat{\beta}_1}{\hat{\sigma}/\sqrt{n}s_X}\right| \\
&\quad -\frac{1}{2n}\frac{\hat{\beta}_1^2}{\hat{\sigma}^2/ns_X^2} \qquad (7.52)\\
&= \frac{\log\sqrt{2}\pi}{n} \\
&\quad +\frac{\log\left|\frac{\hat{\beta}_1}{\hat{\sigma}/s_x}\right|}{n} \\
&\quad -\frac{\log n}{2n} \\
&\quad -\frac{\hat{\beta}_1^2}{2\hat{\sigma}^2/s_X^2} \qquad (7.53)
\end{aligned}$$

Take the limit as $n \rightarrow \infty$:

$$\begin{aligned}
\lim_{n\rightarrow\infty}\frac{1}{n}\log P_n &\leq \lim_{n}\frac{\log\sqrt{2}\pi}{n} \\
&\quad +\lim_{n}\frac{\log\frac{\hat{\beta}_1}{\hat{\sigma}/s_x}}{n} \\
&\quad -\lim_{n}\frac{\log n}{2n} \\
&\quad -\lim_{n}\frac{\hat{\beta}_1^2}{2\hat{\sigma}^2/s_X^2} \qquad (7.54)
\end{aligned}$$

Since $\hat{\beta}_1/(\hat{\sigma}/s_X) \rightarrow \beta_1/(\sigma/s_X)$, and $n^{-1}\log n \rightarrow 0$,

$$\lim_{n\rightarrow\infty}\frac{1}{n}\log P_n \leq -\frac{\beta_1^2}{2\sigma^2/s_X^2} \qquad (7.55)$$

I've only used the upper bound on $1-\Phi(x)$ from Eq. 7.48; if we use the lower bound from that equation, we get (Exercise 5)

$$\lim_{n\rightarrow\infty}\frac{1}{n}\log P_n \geq -\frac{\beta_1^2}{2\sigma^2/s_X^2} \qquad (7.56)$$

Putting the upper and lower limits together,

$$\lim_{n\rightarrow\infty}\frac{1}{n}\log P_n = -\frac{\beta_1^2}{2\sigma^2/s_X^2}$$

Turn the limit around: at least for large n,

$$P_n \approx e^{-n \frac{\beta_1^2}{2\sigma^2/s_X^2}} \tag{7.57}$$

Thus, *any* $\beta_1 \neq 0$ will (eventually) give exponentially small p-values. This is why, as a saying among statisticians have it, "the p-value is a measure of sample size": any non-zero coefficient will become arbitrarily statistically significant with enough data. This is just another way of saying that with enough data, we can (and will) detect even arbitrarily small coefficients, which is what we *want*. The flip-side, however, is that it's just senseless to say that one coefficient is important because it has a really small p-value and another is unimportant because it's got a big p-value. As we can see from Eq. 7.57, the p-value runs together the magnitude of the coefficient ($|\beta_1|$), the sample size (n), the noise around the regression line (σ^2), and how spread out the data is along the x axis (s_X^2), the last of these because they control how precisely we can estimate β_1. Saying "this coefficient must be really important, because we can measure it really precisely" is not smart.

7.7 Confidence Sets and p-Values in R

When we estimate a model with `lm`, R makes it easy for us to extract the confidence intervals of the coefficients:

```
confint(object, level = 0.95)
```

Here `object` is the name of the fitted model object, and `level` is the confidence level; if you want 95% confidence, you can omit that argument. For instance:

```
library(gamair)
data(chicago)
death.temp.lm <- lm(death ~ tmpd, data = chicago)
confint(death.temp.lm)
##                   2.5 %     97.5 %
## (Intercept) 128.8783687 131.035734
## tmpd         -0.3096816  -0.269607
confint(death.temp.lm, level = 0.9)
##                     5 %        95 %
## (Intercept) 129.0518426 130.8622598
## tmpd         -0.3064592  -0.2728294
```

If you want p-values for the coefficients[18], those are conveniently computed as part of the `summary` function:

```
coefficients(summary(death.temp.lm))
##                Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) 129.9570512 0.55022802 236.18763  0.00000e+00
## tmpd         -0.2896443 0.01022089 -28.33845 3.23449e-164
```

[18] And, really, why do you?

Notice how this actually gives us an array with four columns: the point estimate, the standard error, the t statistic, and finally the p-value. Each row corresponds to a different coefficient of the model. If we want, say, the p-value of the intercept, that's

```
coefficients(summary(death.temp.lm))[1, 4]
## [1] 0
```

The summary function will also print out a *lot* of information about the model:

```
summary(death.temp.lm)
##
## Call:
## lm(formula = death ~ tmpd, data = chicago)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -42.275  -9.018  -0.754   8.187 305.952
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 129.95705    0.55023  236.19   <2e-16 ***
## tmpd         -0.28964    0.01022  -28.34   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.22 on 5112 degrees of freedom
## Multiple R-squared:  0.1358,Adjusted R-squared:  0.1356
## F-statistic: 803.1 on 1 and 5112 DF,  p-value: < 2.2e-16
```

As my use of `coefficients(summary(death.temp.lm))` above suggests, the `summary` function actually returns a complex object, which can be stored for later access, and printed. Controlling how it gets printed is done through the `print` function:

```
print(summary(death.temp.lm), signif.stars = FALSE, digits = 3)
##
## Call:
## lm(formula = death ~ tmpd, data = chicago)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -42.27  -9.02  -0.75   8.19 305.95
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 129.9571     0.5502   236.2   <2e-16
## tmpd         -0.2896     0.0102   -28.3   <2e-16
##
## Residual standard error: 14.2 on 5112 degrees of freedom
## Multiple R-squared:  0.136,Adjusted R-squared:  0.136
## F-statistic:  803 on 1 and 5112 DF,  p-value: <2e-16
```

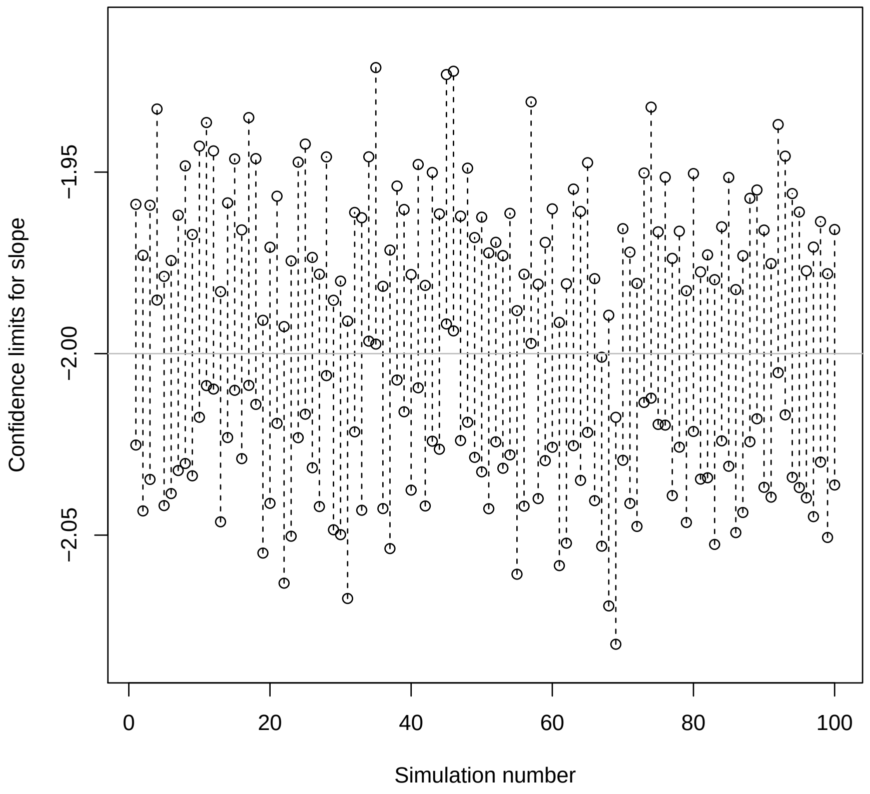
Here I am indulging in two of my pet peeves. It's been conventional (at least since the 1980s) to decorate this sort of regression output with stars beside the coefficients which are significant at various traditional levels. Since (as we've just seen at tedious length) statistical significance has almost nothing to do with real importance, this just clutters the print-out to no advantage[19]. Also, `summary` has a bad habit of using far more significant[20] digits than is justified by the precision of the estimates; I've reined that in.

7.7.1 Coverage of the Confidence Intervals: A Demo

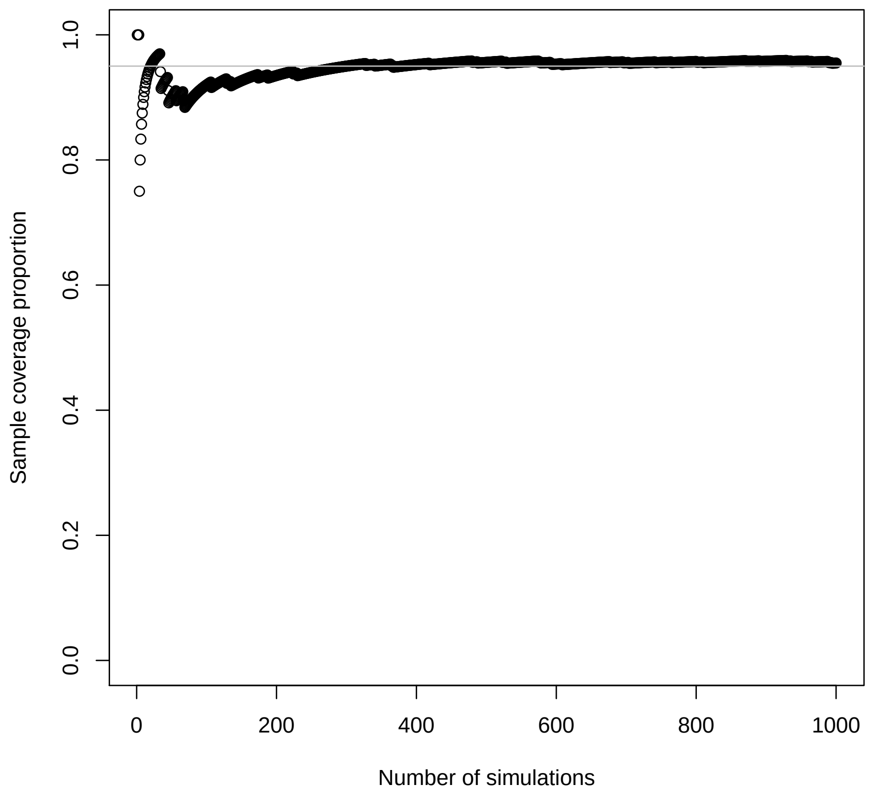
Here is a little computational demonstration of how the confidence interval for a parameter is a random parameter, and how it covers the true parameter value with the probability we want. I'll repeat many simulations of the model from Figure 7.2, calculate the confidence interval on each simulation, and plot those. I'll also keep track of how often, in the first m simulations, the confidence interval covers the truth; this should converge to $1-\alpha$ as m grows.

[19]In fact, I strongly recommend running `options(show.signif.stars=FALSE)` at the beginning of your R script, to turn off the stars forever.

[20]A different sense of "significant"!



```
# Run 1000 simulations and get the confidence interval from each
CIs <- replicate(1000, confint(lm(y ~ x, data = sim.gnslrm(x = x, 5, -2, 0.1,
    FALSE)))[2, ])
# Plot the first 100 confidence intervals; start with the lower limits
plot(1:100, CIs[1, 1:100], ylim = c(min(CIs), max(CIs)), xlab = "Simulation number",
    ylab = "Confidence limits for slope")
# Now the lower limits
points(1:100, CIs[2, 1:100])
# Draw line segments connecting them
segments(x0 = 1:100, x1 = 1:100, y0 = CIs[1, 1:100], y1 = CIs[2, 1:100], lty = "dashed")
# Horizontal line at the true coefficient value
abline(h = -2, col = "grey")
```



```
# For each simulation, check whether the interval covered the truth
covered <- (CIs[1, ] <= -2) & (CIs[2, ] >= -2)
# Calculate the cumulative proportion of simulations where the interval
# contained the truth, plot vs. number of simulations.
plot(1:length(covered), cumsum(covered)/(1:length(covered)), xlab = "Number of simulations",
    ylab = "Sample coverage proportion", ylim = c(0, 1))
abline(h = 0.95, col = "grey")
```

7.8 Further Reading

There is a lot of literature on significance testing and p-values. They are often quite badly abused, leading to a harsh reaction against them, which in some cases goes as badly wrong as the abuses being complained of[21]. I find the work of D. Mayo and collaborators particularly useful here (Mayo, 1996; Mayo and Cox, 2006; Mayo and Spanos, 2006). You may also want to read `http://bactra.org/weblog/1111.html`, particularly if you find §7.6 interesting, or confusing.

The correspondence between confidence sets and hypothesis tests goes back to Neyman (1937), which was the first formal, conscious introduction of confidence sets. (As usual, there are precursors.) That every confidence set comes from inverting a hypothesis test is a classical result in statistical theory, which can be found in, e.g., Casella and Berger (2002). (See also Exercises 2 and 3 below.) Some confusion on this point seems to arise from people not realizing that "does $\hat{\beta}_1$ fall inside the sampling interval for β_1^*?" is a test of the hypothesis that $\beta_1 = \beta_1^*$.

In later chapters, we will look at how to get confidence sets for multiple parameters at once (when we do multiple linear regression), and how to get confidence sets by simulation, without assuming Gaussian noise (when we introduce the bootstrap).

Exercises

1. *Confidence interval for σ^2:* Start with the observation that $n\hat{\sigma}^2/\sigma^2 \sim \chi^2_{n-2}$.
   (a) Find a formula for the $1-\alpha$ sampling interval for $\hat{\sigma}^2$, in terms of the CDF of the χ^2_{n-2} distribution, α, n and σ^2. (Some of these might not appear in your answer.) Is the width of your sampling interval the same for all σ^2, the way the width of the sampling interval for $\hat{\beta}_1$ doesn't change with β_1?
   (b) Fix $\alpha = 0.05$, $n = 40$, and plot the sampling intervals against σ^2.
   (c) Find a formula for the $1-\alpha$ confidence interval for σ^2, in terms of $\hat{\sigma}^2$, the CDF of the χ^2_{n-2} distribution, α and n.
2. Suppose we start a way of testing the hypothesis $\beta = \beta^*$ which can be applied to any β^*, and which has size (false alarm / type I error) probability α for β^*. Show that the set of β retained by their tests is a confidence set, with confidence level $1-\alpha$. What happens if the size is $\leq \alpha$ for all β^* (rather than exactly α)?
3. Suppose we start from a way of creating confidence sets which we know has confidence level $1-\alpha$. We test the hypothesis $\beta = \beta^*$ by rejecting when β^* is outside the confidence set, and retaining when β^* is inside the confidence set. Show that the size of this test is α. What happens if the initial confidence level is $\geq 1-\alpha$, rather exactly $1-\alpha$?

[21]Look, for instance, at the exchange between McCloskey (2002); McCloskey and Ziliak (1996) and Hoover and Siegler (2008).

4. Prove that the p-value P is uniformly distributed under the null hypothesis. You may, throughout, assume that the test statistic T has a continuous distribution.
   (a) Show that if $Q \sim \text{Unif}(0,1)$, then $P = 1 - Q$ has the same distribution.
   (b) Let X be a continuous random variable with CDF F. Show that $F(X) \sim \text{Unif}(0,1)$. *Hint:* the CDF of the uniform distribution $F_{\text{Unif}(0,1)}(x) = x$.
   (c) Show that P, as defined, is $1 - F_{|T|}(|T_{obs}|)$.
   (d) Using the previous parts, show that $P \sim \text{Unif}(0,1)$.
5. Use Eq. 7.48 to show Eq. 7.56, following the derivation of Eq. 7.55.

Chapter 8

Predictive Inference for the Simple Linear Model

There are (at least) three levels at which we can make predictions with a regression model: we can give a single best guess about what Y will be when $X = x$, a *point prediction*; we can try to guess the whole probability distribution of possible Y values, a *distributional* prediction; or we can, less ambitiously, try to make an *interval* prediction, saying "with such-and-probability, Y will be in the interval between here and there".

8.1 Confidence intervals for conditional means

The conditional mean at any particular x is just a number; we can do inference on it as though it were a parameter; it is, after all, a function of the parameters. More specifically, the true conditional mean is

$$m(x) \equiv \mathbb{E}[Y|X = x] = \beta_0 + \beta_1 x \tag{8.1}$$

while our estimate of the conditional mean is

$$\hat{m}(x) = \hat{\beta}_0 + \hat{\beta}_1 x \tag{8.2}$$

(See note on notation below.)

We've seen, in Chapter 3, that

$$\hat{m}(x) = \beta_0 + \beta_1 x + \frac{1}{n}\sum_{i=1}^{n}\left(1 + (x - \overline{x})\frac{x_i - \overline{x}}{s_X^2}\right)\epsilon_i \tag{8.3}$$

so that

$$\mathbb{E}[\hat{m}(x)] = \beta_0 + \beta_1 x = m(x) \tag{8.4}$$

and

$$\mathrm{Var}[\hat{m}(x)] = \frac{\sigma^2}{n}\left(1 + \frac{(x - \overline{x})^2}{s_X^2}\right) \tag{8.5}$$

Under the Gaussian noise assumption, $\hat{m}(x)$ is Gaussian (why?),

$$\hat{m}(x) \sim N\left(m(x), \frac{\sigma^2}{n}\left(1 + \frac{(x-\overline{x})^2}{s_X^2}\right)\right) \tag{8.6}$$

Notice how the variance grows as we move further and further away from the center of the data along the x axis. Also notice how all the unknown parameters show up on the right-hand side of the equation.

Exact confidence intervals At this point, getting confidence intervals for $m(x)$ works just like getting confidence intervals for β_0 or β_1: we use as our standard error

$$\widehat{\mathrm{se}}\left[\hat{m}(x)\right] = \frac{\hat{\sigma}}{\sqrt{n-2}}\sqrt{1 + \frac{(x-\overline{x})^2}{s_X^2}} \tag{8.7}$$

and then find

$$\frac{\hat{m}(x) - m(x)}{\widehat{\mathrm{se}}\left[\hat{m}(x)\right]} \sim t_{n-2} \tag{8.8}$$

by entirely parallel arguments. $1-\alpha$ confidence intervals follow as before as well.

What about using CIs for β_0 and β_1? That's not a bad idea, but since $m(x)$ is a function of both parameters, we'd need a simultaneous confidence region, not two confidence intervals. Similarly, we could try using the sampling distributions of $\hat{\beta}_0$ and $\hat{\beta}_1$ to get the distribution of $\hat{m}(x)$, but then we need to worry about the covariance between them. Eq. 8.3 effectively handles all those awkward complications for us, by breaking $\hat{m}(x)$ down into its component parts.

Notation The textbook, following an old tradition, talks about $\hat{y}$ as the conditional mean. This is not a good tradition, since it leads to great awkwardness in distinguishing the true conditional mean from our estimate of it. Hence my use of $m(x)$ and $\hat{m}(x)$.

8.1.1 Interpreting the confidence interval

This confidence interval has the same interpretation as one for the parameters: either

1. The true value of $m(x)$, i.e., the true value of $\mathbb{E}[Y|X=x]$, is in the interval, or
2. Something very unlikely happened when we got our data.

This is all well and good, but it does not tell us about how often future values of Y will be in this interval; it tells us about how often we capture the conditional average.

8.1.2 Large-n approximation

As n grows, the t distribution with $n-2$ degrees of freedom becomes, approximately, the standard Gaussian. It follows that for large n,

$$\frac{\hat{m}(x) - m(x)}{\widehat{\text{se}}[\hat{m}(x)]} \approx N(0,1) \tag{8.9}$$

so

$$\hat{m}(x) \approx N(m(x), \widehat{\text{se}}[\hat{m}(x)]^2) \tag{8.10}$$

and an approximate $1-\alpha$ confidence interval for $m(x)$ is

$$\hat{m}(x) \pm z_{\alpha/2} \frac{\hat{\sigma}}{\sqrt{n}} \sqrt{1 + \frac{(x-\overline{x})^2}{s_X^2}} \tag{8.11}$$

(It doesn't matter whether we use $n-2$ or n in the denominator for $\widehat{\text{se}}$.) Notice that the width of this interval $\rightarrow 0$ as $n \rightarrow \infty$.

8.1.3 Confidence intervals and transformations

Transforming the predictor variable raises no particular issues. Transforming the response, however, is quite delicate.

When we transform the response, the model becomes

$$g(Y) = \beta_0 + \beta_1 x + \epsilon \tag{8.12}$$

for ϵ IID Gaussian, $N(0, \sigma^2)$. Now

$$\mathbb{E}[g(Y) \mid X = x] = \beta_0 + \beta_1 x \tag{8.13}$$

and so if we go through the calculates above, we get confidence intervals for $\mathbb{E}[g(Y) \mid X = x]$, the conditional expectation of the *transformed* response.

In general, however,

$$\mathbb{E}[Y \mid X = x] \neq g^{-1}(\beta_0 + \beta_1 x) \tag{8.14}$$

so just applying g^{-1} to the confidence limits for $\mathbb{E}[g(Y) \mid X = x]$ won't give us a confidence interval for $\mathbb{E}[Y|X = x]$.

8.2 Prediction Interval

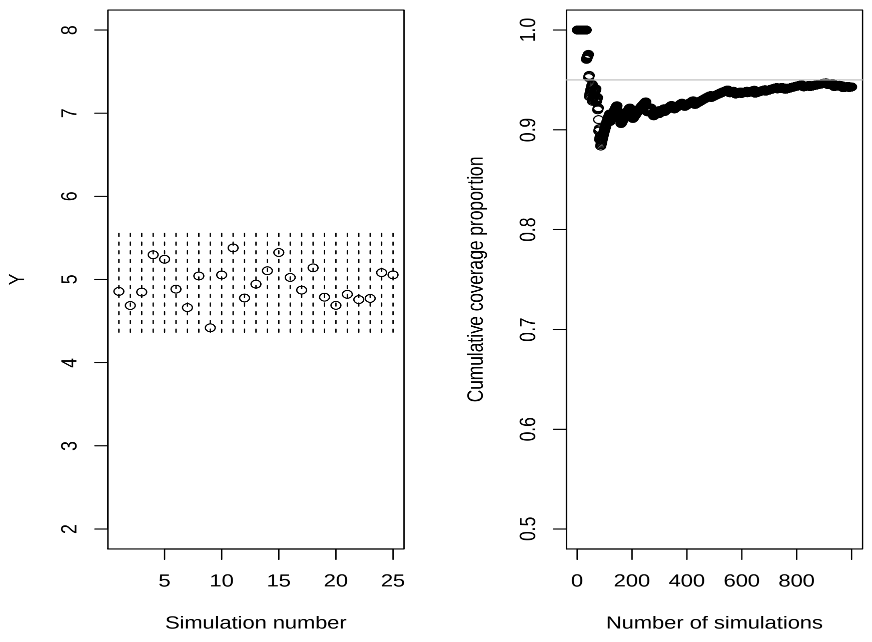
A $1-\alpha$ **prediction interval** for $Y|X = x$ is a an interval $[l, u]$ where

$$\mathbb{P}(l \leq Y \leq u|X = x) = 1 - \alpha \tag{8.15}$$

Since $Y|X = x \sim N(m(x), \sigma^2)$, it would be a simple matter to find these limits if we knew the parameters: the lower limit would be $m(x) + z_{\alpha/2}\sigma$, and the upper limit $m(x) + z_{1-\alpha/2}\sigma$. Unfortunately, we don't know the parameters.

```
# Simulate a Gaussian-noise simple linear regression model Inputs: x
# sequence; intercept; slope; noise variance; switch for whether to return
# the simulated values, or run a regression and return the estimated model
# Output: data frame or coefficient vector
sim.gnslrm <- function(x, intercept, slope, sigma.sq, mdl = TRUE) {
    n <- length(x)
    y <- intercept + slope * x + rnorm(n, mean = 0, sd = sqrt(sigma.sq))
    if (mdl) {
        return(lm(y ~ x))
    } else {
        return(data.frame(x = x, y = y))
    }
}

# Read in a model and get it to give a prediction interval at a given x This
# will be convenient when we want to have lots of models make predictions at
# the same point Inputs: the model, the value of x Output: vector giving
# prediction interval
extract.pred.int <- function(mdl, x, level = 0.95) {
    predict(mdl, newdata = data.frame(x = x), interval = "prediction", level = level)
}


# No magic numbers!
x.new <- 1/137
m <- 1000
alpha <- 0.05
beta.0 <- 5
beta.1 <- -2
sigma.sq <- 0.1
```

FIGURE 8.1: *Code setting up a simulation of a Gaussian-noise simple linear regression model, along a fixed vector of x_i values, followed by some default values we'll use in the later simulations.*



```
# Simulate Y from the model
y.new <- sim.gnslrm(x = rep(x.new, m), beta.0, beta.1, sigma.sq, mdl = FALSE)$y
# All the prediction intervals are the same (because x isn't changing)
pred.int <- beta.0 + beta.1 * x.new + sqrt(sigma.sq) * qnorm(c(alpha/2, 1 -
    alpha/2))
names(pred.int) <- c("lwr", "upr")  # Names make for clearer code
par(mfrow = c(1, 2))  # Set up for 2 side-by-side plots
# Plot the first 25 runs of Y (so we can see what's happening)
plot(1:25, y.new[1:25], xlab = "Simulation number", ylab = "Y", ylim = c(2,
    8))
# Add vertical segments for the prediction intervals
segments(x0 = 1:25, x1 = 1:25, y0 = pred.int["lwr"], y1 = pred.int["upr"], lty = "dashed")
# For each Y, check if it's covered by the interval
covered <- (y.new >= pred.int["lwr"]) & (y.new <= pred.int["upr"])
# Plot the running mean of the fraction of Y's covered by the interval
plot(1:m, cumsum(covered)/(1:m), xlab = "Number of simulations", ylab = "Cumulative coverage proportion",
    ylim = c(0.5, 1))
abline(h = 1 - alpha, col = "grey")  # Theoretical coverage level
par(mfrow = c(1, 1))  # Restore ordinary plot layout for later
```

FIGURE 8.2: *Demonstration of the coverage of the prediction intervals. Here, we are seeing what would happen if we got to use the true coefficients, which are $\beta_0 = 5$, $\beta_1 = -2$, $\sigma^2 = 0.1$; we are always trying to predict Y when $X = 1/137$.*

However, we do know how the parameters are related to our estimates, so let's try to use that:

$$\begin{aligned} Y|X=x &\sim N(m(x),\sigma^2) && (8.16)\\ &= m(x)+N(0,\sigma^2) && (8.17)\\ &= \hat{m}(x)+N\left(0,\frac{\sigma^2}{n}\left(1+\frac{(x-\overline{x})^2}{s_X^2}\right)\right)+N(0,\sigma^2) && (8.18)\\ &= \hat{m}(x)+N\left(0,\sigma^2\left(1+\frac{1}{n}+\frac{(x-\overline{x})^2}{ns_X^2}\right)\right) && (8.19) \end{aligned}$$

where in the last line I've used the fact that, under the assumptions of the model, the new Y we're trying to predict is independent of the old Y used to estimate the parameters. The variance, as we've seen, has two parts: the true noise variance about the regression line, plus the variance coming from our uncertainty in where that regression line is. Both parts of the variance are proportional to σ^2. Let's call the whole thing $\sigma^2_{pred}(x)$.

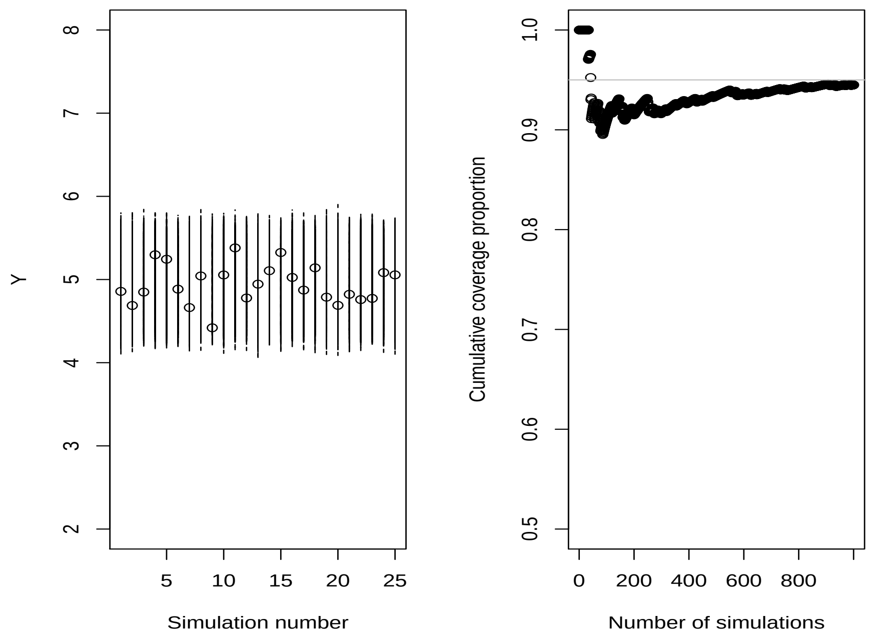
So, we have a random variable with a Gaussian distribution centered at $\hat{m}(x)$ and with a variance $\sigma^2_{pred}(x)$ proportional to σ^2. We can estimate that variance as

$$s^2_{pred}(x)=\hat{\sigma}^2\frac{n}{n-2}\left(1+\frac{1}{n}+\frac{(x-\overline{x})^2}{ns_X^2}\right) \tag{8.20}$$

Going through the now-familiar argument once again,

$$\frac{Y-\hat{m}(x)}{s_{pred}(x)}\mid X=x\sim t_{n-2} \tag{8.21}$$

and we can use this to give prediction intervals.



```
# Run simulations where we get a new estimate of the model on each run, but
# with fixed X vector (to keep it simple)
x.seq <- seq(from = -5, to = 5, length.out = 42)
# Run the simulation many times, and give a _list_ of estimated models
# simplify=FALSE forces the return value to be a list
mdls <- replicate(m, sim.gnslrm(x = x.seq, beta.0, beta.1, sigma.sq, mdl = TRUE),
    simplify = FALSE)
# Extract the prediction intervals for every one of the models
pred.ints <- sapply(mdls, extract.pred.int, x = x.new)
rownames(pred.ints)[2:3] <- names(pred.int)  # Fix the names
# Now make plots like the previous figure
par(mfrow = c(1, 2))
plot(1:25, y.new[1:25], xlab = "Simulation number", ylab = "Y", ylim = c(2,
    8))
segments(x0 = 1:25, x1 = 1:25, y0 = pred.ints["lwr", ], y1 = pred.ints["upr",
    ], lty = "dashed")
covered <- (y.new >= pred.ints["lwr", ]) & (y.new <= pred.ints["upr", ])
plot(1:m, cumsum(covered)/(1:m), xlab = "Number of simulations", ylab = "Cumulative coverage proportion",
    ylim = c(0.5, 1))
abline(h = 1 - alpha, col = "grey")
par(mfrow = c(1, 1))
```

FIGURE 8.3: *As in Figure 8.2, but we are now using coefficients estimated by drawing 42 observations from the model, with the X's being evenly spaced from* -5 *to* 5*. Here, as you can see from the code, each prediction is made on the basis of a different random realization of the data before estimating the model. (See §8.3 below for details on how to use* `predict` *to return intervals.)*

Again, as usual, as $n \to \infty$, the t distribution turns into a standard Gaussian, while $s^2_{pred}(x) \to \sigma^2_{pred}(x) \to \sigma^2$. With *enough* data, then, our prediction intervals approach the ones we'd use if we knew the parameters and they were exactly our point estimates. Notice that the width of these prediction intervals does *not* go to zero as $n \to \infty$ — there is always some noise around the regression line!

8.2.1 Interpretation of the prediction interval

The interpretation of the prediction interval here is a bit tricky.

What we want for a prediction interval is that

$$\mathbb{P}(l \leq Y \leq u \mid X = x) = 1 - \alpha \tag{8.22}$$

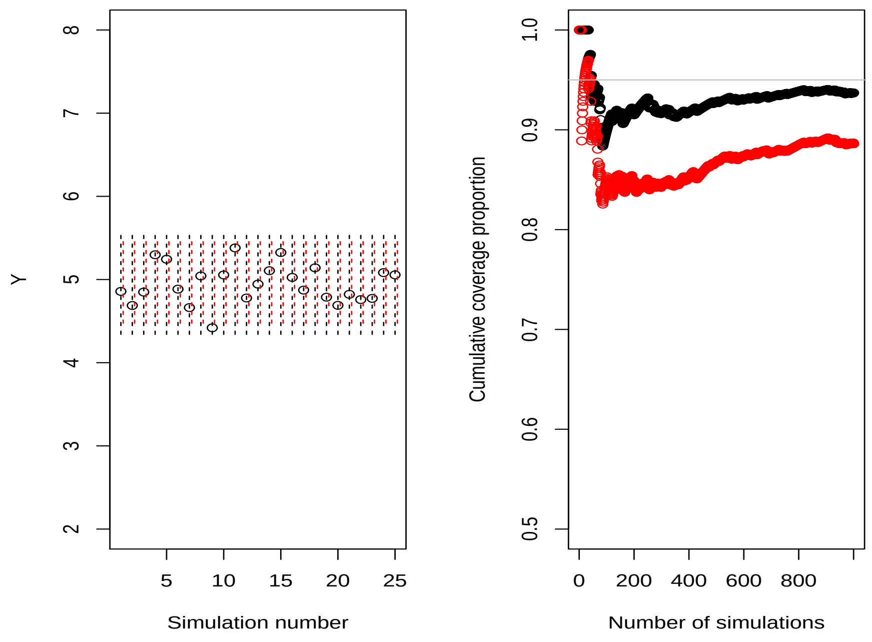
Now our limits l and u involve the estimated parameters. To be explicit,

$$\mathbb{P}\left(\hat{m}(x) + t_{n-2}(\alpha/2)s_{pred}(x) \leq Y \leq \hat{m}(x) + t_{n-2}(1-\alpha/2)s_{pred}(x) \mid X = x\right) = 1 - \alpha \tag{8.23}$$

But $\hat{m}(x)$ and $s_{pred}(x)$ are both random variables. The experiment we're imagining repeating when we write out Eq. 8.23 involves both estimating the parameters and predicting a new Y at $X = x$ every time.

If we estimate the parameters just once, and then try repeatedly measuring Y when $X = x$, we'll see that our coverage level, while close to $1-\alpha$, is not quite $1-\alpha$, sometimes less and sometimes more. (But over many estimates, the coverage must average out to $1 - \alpha$ — why?) The coverage gets closer to the desired level as the number of points n used to *estimate* the model grows, but simply predicting more observations with fixed estimates won't budge it.

It is nonetheless a Bad Sign for the model if the actual coverage level is very far from $1-\alpha$, especially if the coverage for certain regions of the x axis is very far from this desired or nominal level. One might, however, need to do some simulations (along the lines of the code provided here...) to see how big a departure should be expected if all the model assumptions hold.



```
# What's the coverage if we use just one estimate of the model?  Pick the
# first two, arbitrarily, to show how this varies Get the prediction
# interval for our x.new
pred.ints <- sapply(mdls[1:2], extract.pred.int, x = x.new)
rownames(pred.ints)[2:3] <- c("lwr", "upr")
# Make the plots
par(mfrow = c(1, 2))
plot(1:25, y.new[1:25], xlab = "Simulation number", ylab = "Y", ylim = c(2,
    8))
segments(x0 = 1:25, x1 = 1:25, y0 = pred.ints["lwr", 1], y1 = pred.ints["upr",
    1], lty = "dashed")
# Slightly off-set one of the intervals for visibility
segments(x0 = 0.2 + 1:25, x1 = 0.2 + 1:25, y0 = pred.ints["lwr", 2], y1 = pred.ints["upr",
    2], lty = "dashed", col = "red")
# Calculate two cumulative coverage proportions
covered.1 <- (y.new >= pred.ints["lwr", 1]) & (y.new <= pred.ints["upr", 1])
covered.2 <- (y.new >= pred.ints["lwr", 2]) & (y.new <= pred.ints["upr", 2])
plot(1:m, cumsum(covered.1)/(1:m), xlab = "Number of simulations", ylab = "Cumulative coverage proportion",
    ylim = c(0.5, 1))
points(1:m, cumsum(covered.2)/(1:m), col = "red")
abline(h = 1 - alpha, col = "grey")
par(mfrow = c(1, 1))
```

FIGURE 8.4: *As in Figure 8.3, but all the new realizations of Y are being predicted based on the coefficients of one single estimate of the coefficients (the first estimate for the black intervals, the second estimate for the red). — The code for all three figures is very similar; could you write one function which, with appropriate arguments, would make all three of them?*

8.2.2 Prediction intervals and transformations

Transforming the predictor variable raises no issues for prediction intervals. If we've transformed the response, though, we need to take account of it.

A model with a transformed response looks like this:

$$g(Y) = \beta_0 + \beta_1 X + \epsilon \tag{8.24}$$

for ϵ IID Gaussian, and some invertible, non-linear function g. Since g is invertible, it must be either increasing or decreasing; to be definite, I'll say it's increasing, but it should be clear as we go what needs to change for decreasing transformations.

When we estimated the model after transforming Y, what we have above gives us a prediction interval for $g(Y)$. Remember what this means:

$$\mathbb{P}(L \leq g(Y) \leq U|X = x) = 1 - \alpha \tag{8.25}$$

Since g is an increasing function, so is g^{-1}, and therefore

$$\{L \leq g(Y) \leq U\} \Leftrightarrow \left\{g^{-1}(L) \leq Y \leq g^{-1}(U)\right\} \tag{8.26}$$

Since the two events are logically equivalent, they must have the same probability, no matter what we condition on:

$$\mathbb{P}\left(g^{-1}(L) \leq Y \leq g^{-1}(U) \,|\, X = x\right) = 1 - \alpha \tag{8.27}$$

Thus, we get a prediction interval for Y by taking the prediction interval for $g(Y)$ and undoing the transformation.

8.3 Prediction intervals in R

For linear models, all of the calculations needed to find confidence intervals for $\hat{m}$ or prediction intervals for Y are automated into the `predict` function, introduced in Chapter 4.

```
predict(object, newdata, interval = c("none", "confidence", "prediction"), level = 0.95)
```

The `object` argument is the estimated model returned by `lm`; `newdata` is a data frame containing a column whose name matches that of the predictor variable. We saw these arguments before (see Chapter 4 again); what's new are the other two. `interval` controls whether to give point predictions (`"none"`, the default) or intervals, and if so which kind. `level` is of course the confidence level (default 0.95 for tradition's sake.)

To illustrate, let's revisit our old friend `chicago`:

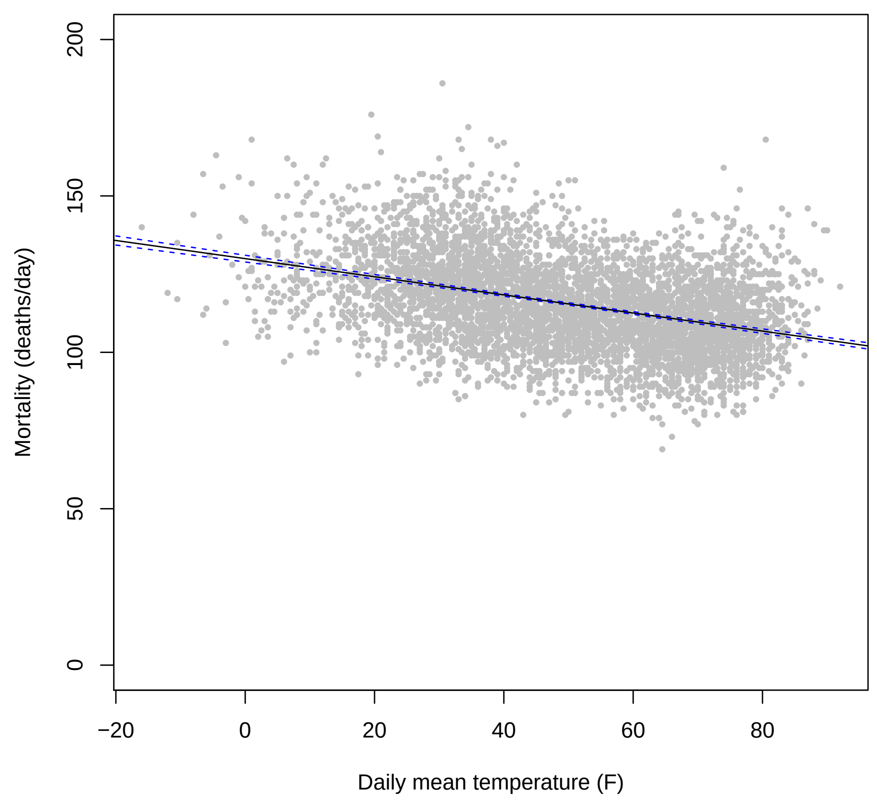
```
library(gamair)
data(chicago)
death.temp.lm <- lm(death ~ tmpd, data = chicago)
```

Figure 8.5 shows a scatter-plot of the data and the estimated line, together with confidence limits for the conditional mean at each point[1]. Because we have thousands of data points and reasonably large s_X^2, the confidence limits are quite narrow, though you can see, from the plot, how they widen as we move away from the mean temperature[2].

Figure 8.6 shows the *prediction* limits for the same model. These are much wider, because their width is mostly coming from (the estimate of) σ, the noise around the regression line, the model being very confident that it knows what the line is. Despite their width, the bands don't include all the data points. This is not, in itself, alarming — they should only contain about 95% of the data points! I will leave it as an exercise to check what the actual coverage level is here.

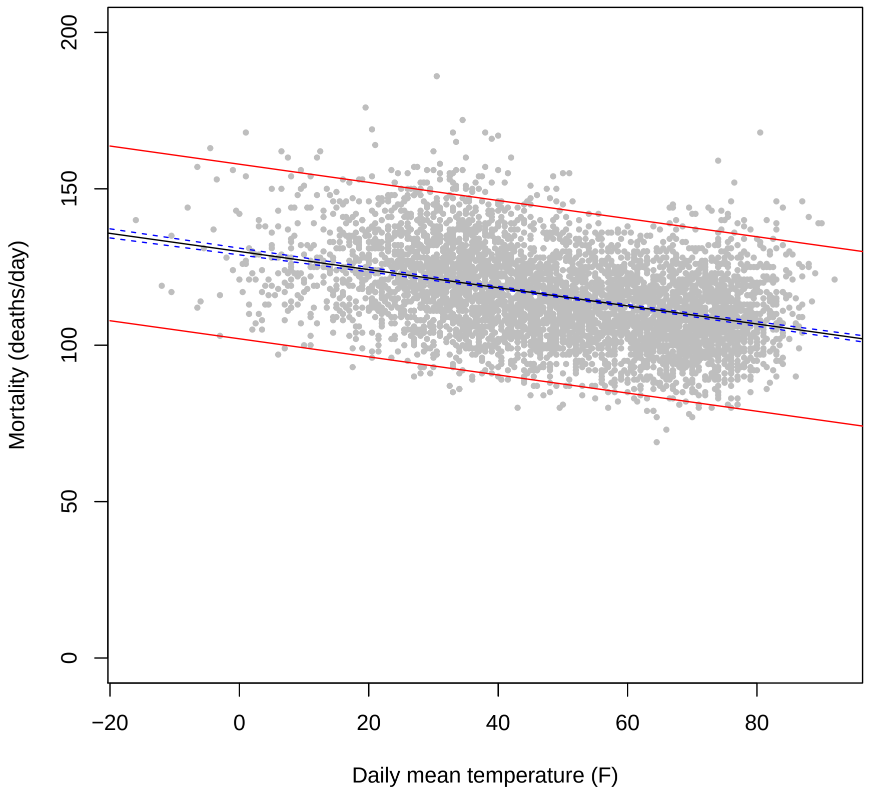
[1]The confidence limits we've worked out are for $m(x)$ at a specific x. If we wanted curves $L(x)$, $U(x)$ which would bracket $m(x)$ everywhere with high probability (i.e., $\mathbb{P}(\forall x,\ L(x) \leq m(x) \leq U(x)) = 1-\alpha$, we need a slightly more complicated construction.

[2]You'll notice that they don't widen enough to include the non-parametric (spline) estimate of the conditional mean. This is another sign that the model is making systematic mistakes at high temperatures.



```
plot(death ~ tmpd, data = chicago, pch = 19, cex = 0.5, col = "grey", ylim = c(0,
    200), xlab = "Daily mean temperature (F)", ylab = "Mortality (deaths/day)")
abline(death.temp.lm)
temp.seq <- seq(from = -20, to = 100, length.out = 100)
death.temp.CIs <- predict(death.temp.lm, newdata = data.frame(tmpd = temp.seq),
    interval = "confidence")
lines(temp.seq, death.temp.CIs[, "lwr"], lty = "dashed", col = "blue")
lines(temp.seq, death.temp.CIs[, "upr"], lty = "dashed", col = "blue")
```

FIGURE 8.5: *Data from the Chicago death example (grey dots), together with the regression line (solid black) and the 95% confidence limits on the conditional mean (dashed blue curves). I have restricted the vertical range to help show the confidence limits, though this means some high-mortality days are off-screen.*



```
plot(death ~ tmpd, data = chicago, pch = 19, cex = 0.5, col = "grey", ylim = c(0,
    200), xlab = "Daily mean temperature (F)", ylab = "Mortality (deaths/day)")
abline(death.temp.lm)
temp.seq <- seq(from = -20, to = 100, length.out = 100)
death.temp.CIs <- predict(death.temp.lm, newdata = data.frame(tmpd = temp.seq),
    interval = "confidence")
lines(temp.seq, death.temp.CIs[, "lwr"], lty = "dashed", col = "blue")
lines(temp.seq, death.temp.CIs[, "upr"], lty = "dashed", col = "blue")
death.temp.PIs <- predict(death.temp.lm, newdata = data.frame(tmpd = temp.seq),
    interval = "prediction")
lines(temp.seq, death.temp.PIs[, "lwr"], col = "red")
lines(temp.seq, death.temp.PIs[, "upr"], col = "red")
```

FIGURE 8.6: *Adding 95% prediction intervals (red) to the previous plot.*

Chapter 9

Interpreting Parameters after Transformation

9.1 Transformed Predictor

The model becomes

$$Y = \beta_0 + \beta_1 f(X) + \epsilon \tag{9.1}$$

for some invertible, nonlinear function f, with ϵ still IID Gaussian. The usual interpretations apply, but now are all in terms of $f(x)$, not x:

1. We can never find coefficients γ_0, γ_1 where

$$\beta_0 + \beta_1 f(x) = \gamma_0 + \gamma_1 x \tag{9.2}$$

   for all x. That is to say, applying a nonlinear transformation to the predictor doesn't just amount to making some adjustment to the slope and intercept.

2. $\beta_0 = \mathbb{E}[Y|f(X) = 0]$. This is (usually) not $\mathbb{E}[Y|X = 0]$.
   (a) β_0 is still the intercept when $f(x)$ goes on the horizontal axis.
   (b) Instead, $\mathbb{E}[Y|X = 0] = \beta_0 + \beta_1 f^{-1}(0)$.

3. β_1 is the slope in units of Y per units of $f(X)$. That is, it's the difference in the expected response for a difference in $f(x)$ of 1, not for a difference in x of 1.
   (a) A difference of 1 in x predicts a difference of $\beta_1(f(x) - f(x-1))$ in $\mathbb{E}[Y]$ if x decreases by 1, and a difference of $\beta_1(f(x+1) - f(x))$ if x increases by 1. (These are generally not the same.) So even the response to increases and decreases isn't necessarily of the same size.
   (b) Very small differences in x, of size h, predict very small differences in $\mathbb{E}[Y]$, of size $h\beta_1 \frac{df}{dx}(x)$. So there is a slope at each point, but it changes. (That's what makes f non-linear.)

4. σ^2 is still the variance of the Gaussian noise around the regression curve, but now that curve really is curved and not a straight line.
   (a) A plot of y_i against x_i should not be a straight line.
   (b) A plot of $e_i = y_i - (\hat{\beta}_0 + \hat{\beta}_1 f(x_i))$ should still be a flat line around 0.

We estimate the model by transforming the data, going from $(x_1, y_1), \ldots (x_n, y_n)$ to $(f(x_1), y_1), \ldots (f(x_n), y_n)$, and then running a regression of y_i on $f(x_i)$.

9.1.1 Special case: Log transformation of the predictor

Suppose we select $f = \log$. Our model then is

$$Y = \beta_0 + \beta_1 \log X + \epsilon \tag{9.3}$$

In this setting,

- $\log X = 0$ means $X = 1$, so $\beta_0 = \mathbb{E}[Y|X = 1]$.
- A k unit change in $\log x$ means multiplying x by e^k:

  $$k + \log x = \log e^k + \log x = \log x e^k \tag{9.4}$$

  Hence, β_1 is the expected difference in Y for an e-fold change in X.
- The slope of $\mathbb{E}[Y]$ with respect to X decreases in x:

  $$\frac{d\mathbb{E}[Y|X = x]}{dx} = \frac{\beta_1}{x} \tag{9.5}$$

9.2 Transforming the response

Again, we select an invertible, non-linear function g, and transform the response variable:

$$g(Y) = \beta_0 + \beta_1 X + \epsilon \tag{9.6}$$

All of the parameters retain their old interpretations in terms of $g(Y)$. None of them have their old interpretations in terms of Y. This is because the model for Y is now

$$Y = g^{-1}(\beta_0 + \beta_1 X + \epsilon) \tag{9.7}$$

1. We can never find coefficients γ_0, γ_1 where

   $$\beta_0 + \beta_1 x = g(\gamma_0 + \gamma_1 x) \tag{9.8}$$

   for all x. That is to say, applying a nonlinear transformation to the response doesn't just amount to making some adjustment to the slope and intercept.
2. $\beta_0 = \mathbb{E}[g(Y)|X = 0]$. Note that since g is not a linear function, neither is g^{-1}, and so $\mathbb{E}[Y|X = 0] \neq g^{-1}(\beta_0)$.

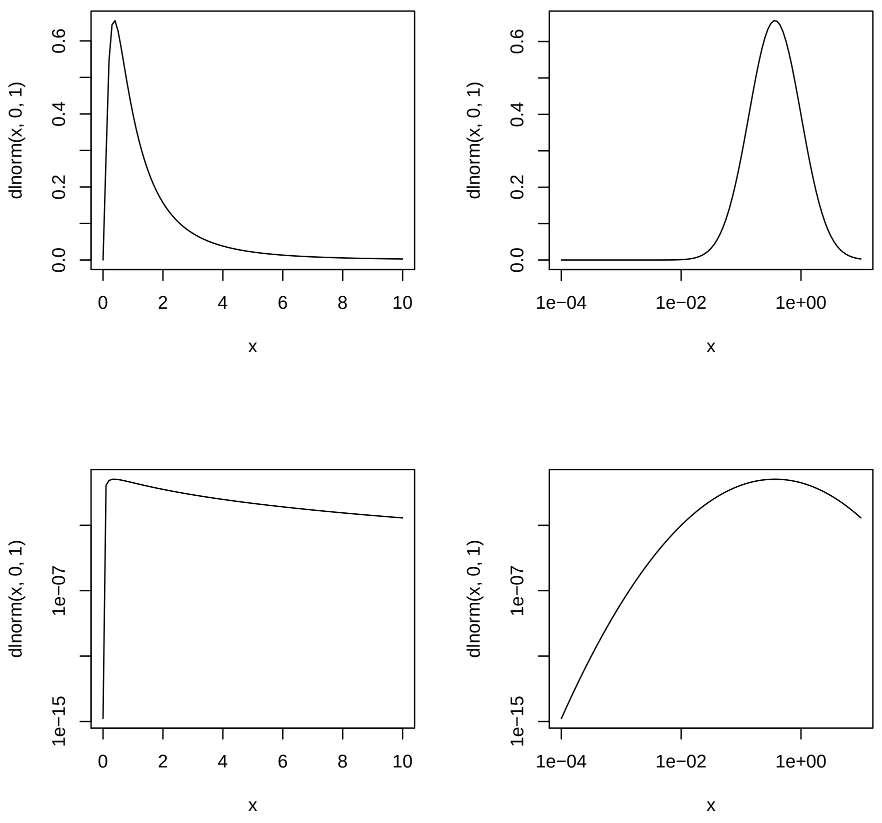
3. More generally, $\mathbb{E}[Y|X=x] \neq g^{-1}(\beta_0+\beta_1 x)$.
   (a) Since we're assuming ϵ is Gaussian centered at 0, the median value of $\epsilon = 0$. Therefore, according to the model,
   $$\mathbb{P}(g(Y) \leq \beta_0+\beta_1 x \,|\, X=x) = 0.5 \tag{9.9}$$
   Since g is invertible, it is therefore also true that
   $$\mathbb{P}\big(Y \leq g^{-1}(\beta_0+\beta_1 x) \,|\, X=x\big) = 0.5 \tag{9.10}$$
   and the transformation can be simply undone for the conditional median, but not the conditional mean.
   (b) In particular, $g^{-1}(\beta_0)$ is the conditional median of Y when $X=0$.
4. β_1 is the difference in the mean of $g(Y)$ predicted by a 1 unit change in X.
   (a) There is generally no simple interpretation of β_1 for the original Y.
   (b) By the argument above, increasing x by h predicts that the conditional *median* will change by $g^{-1}(\beta_0+\beta_1 x+\beta_1 h) - g^{-1}(\beta_0+\beta_1 x)$. This, generally speaking, does not simplify.
   (c) When the change h is very small, the change to the conditional median will tend towards $h\beta_1 \left.\frac{dg^{-1}(u)}{du}\right|_{u=\beta_0+\beta_1 x}$.
5. σ^2 is the variance of the Gaussian noise around the regression line for $g(Y)$.
   (a) Because g^{-1} is a nonlinear transformation, the noise around the regression curve for Y, $g^{-1}(\beta_0+\beta_1 x)$, will not (in general) be Gaussian.
   (b) In fact, the noise around that curve will generally not have mean zero, or constant variance, or even just be an additive perturbation around the curve. (See the example for the log transformation below.)
   (c) A plot of Y against X will not show a straight line, though a plot of $g(Y)$ against X should.
   (d) Residuals for Y, calculated as $y_i - g^{-1}(\hat{\beta}_0+\hat{\beta}_1 x_i)$ need not be centered at 0, or have constant variance, etc., etc.
   (e) Residuals for the transformed response, calculated as $g(y_i)-(\hat{\beta}_0+\hat{\beta}_1 x_i)$, should show all the usual properties.

9.2.1 Special case: log transformation of the response

If we select $g = \log$, the model becomes
$$\log Y = \beta_0+\beta_1 X+\epsilon \tag{9.11}$$
and the interpretation simplifies slightly, especially on taking the inverse transformation:
$$Y = e^{\beta_0+\beta_1 X+\epsilon} = e^{\beta_0} e^{\beta_1 x} e^{\epsilon} \tag{9.12}$$

1. e^{β_0} is the median value of Y when $X = 0$. It is common to abbreviate it as a single number, say y_0.
2. A one-unit increase in x predicts that Y should be larger by a *factor* of e^{β_1}. That is, additive, equal-size changes to x lead to multiplicative changes in Y.
3. The slope of Y with respect to x is $\beta_1 e^{\beta_1 x}$, so, again, there is no one answer to "what gets added to Y when x changes a little?"
4. Because $\epsilon \sim N(0, \sigma^2)$, e^ϵ is not Gaussian (no matter what mean and variance we might try). Rather, we say that e^ϵ is **log-normal** or **log-gaussian**, because it's log is normal or Gaussian[1] (R functions: `dlnorm`, `plnorm`, `qlnorm`, `rlnorm`). This is written $e^\epsilon \sim LN(0, \sigma^2)$, i.e., the log-normal is parameterized by the mean and variance of its log.
   (a) $e^\epsilon \geq 0$, so the LN distribution is supported on the positive numbers, not (like the N) the whole number line.
   (b) It further follows that $\mathbb{E}[e^\epsilon] > 0$.
   (c) By the argument above, the median of $e^\epsilon = 1$.
   (d) Because making $\epsilon < 0$ can only decrease e^ϵ a little below 1 (at worst to 0), but making $\epsilon > 0$ can increase e^ϵ by a lot (up to ∞), the distribution is skewed to the right.
   (e) By directly using the transformation-of-variables formula (which you remember from your probability class), the probability density function of an $LN(\mu, \sigma^2)$ distribution is
   $$\frac{1}{x\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\frac{(\mu - \log x)^2}{\sigma^2}} \tag{9.13}$$
   (Figure 9.1). By integration, then, one can show that the expectation of this distribution is $e^{\mu+\sigma^2/2}$, and its variance is $e^{2\mu+\sigma^2}(e^{\sigma^2} - 1)$.
   (f) Specifically, $\mathbb{E}[e^\epsilon] = \exp \sigma^2/2$.
   (g) Similarly, $\mathrm{Var}[e^\epsilon] = e^{\sigma^2}(e^{\sigma^2} - 1)$.
   (h) The noise e^ϵ *multiplies* the deterministic function $e^{\beta_0+\beta_1 x}$, it does not add to it. Therefore we have
   $$\begin{aligned}
   \mathbb{E}[Y|X=x] &= \mathbb{E}\left[e^{\beta_0+\beta_1 x+\epsilon} \,|\, X=x\right] && (9.14)\\
   &= e^{\beta_0} e^{\beta_1 x}\mathbb{E}[e^\epsilon \,|\, X=x] && (9.15)\\
   &= e^{\beta_0} e^{\beta_1 x}\mathbb{E}[e^\epsilon] && (9.16)\\
   &= e^{\sigma^2/2} e^{\beta_0} e^{\beta_1 x} && (9.17)\\
   \mathrm{Var}[Y|X=x] &= \mathrm{Var}\left[e^{\beta_0+\beta_1 x} e^\epsilon \,|\, X=x\right] && (9.18)\\
   &= e^{2(\beta_0+\beta_1 x)}\mathrm{Var}[e^\epsilon \,|\, X=x] && (9.19)\\
   &= e^{\sigma^2}(e^{\sigma^2} - 1)e^{2(\beta_0+\beta_1 x)} && (9.20)
   \end{aligned}$$

[1]Some people prefer to call e^ϵ "anti-log-normal", which has a kind of logic to it, but they're very much a minority.



```
par(mfrow = c(2, 2))  # Set up 2x2 plotting grid
curve(dlnorm(x, 0, 1), from = 1e-04, to = 10)
curve(dlnorm(x, 0, 1), from = 1e-04, to = 10, log = "x")
curve(dlnorm(x, 0, 1), from = 1e-04, to = 10, log = "y")
curve(dlnorm(x, 0, 1), from = 1e-04, to = 10, log = "xy")
par(mfrow = c(1, 1))  # Restore usual plot playout for later
```

FIGURE 9.1: *Probability density function of the standard log-normal distribution, $LN(0, 1)$. Top left: ordinary (linear) scale on both axes; top right: log scale on horizontal axis; bottom left: log scale on vertical axis; bottom right: log scale on both axes.*

9.3 Transforming both predictor and response

The model is

$$g(Y) = \beta_0 + \beta_1 f(X) + \epsilon \tag{9.21}$$

All the considerations of both the previous sections apply.

1. If this model applies, no linear model also applies.
2. $\beta_0 = \mathbb{E}[g(Y)|f(X) = 0]$.
3. β_1 is the slope of the curve of $g(Y)$ against $f(X)$.
   (a) There is generally no simple way to express this in terms of the original variables.
   (b) There is also generally no simple way to write the slope of the curve of Y on X.
4. σ^2 is the variance of the Gaussian noise around the line of $g(Y)$ against $f(X)$. The distribution of Y around its curve against $f(X)$, let alone against X, is generally not even additive.
5. The function $g^{-1}(\beta_0 + \beta_1 f(x))$ continues to give the conditional median of Y.

9.3.1 Special case: log of both predictor and response

The model is

$$\log Y = \beta_0 + \beta_1 \log X + \epsilon \tag{9.22}$$

Undo the log on both sides:

$$Y = e^{\beta_0} X^{\beta_1} e^{\epsilon} \tag{9.23}$$

Because this says that Y is some power of X, up to noise, this sort of model is often called a **power law**.

Abbreviate e^{β_0} by y_0. Then, in the power law model with log-normal noise,

1. y_0 is the median value of Y when $X = 1$.
2. β_1 is the slope of $\log Y$ against $\log X$. It is also power to which we raise X to get the systematic part of Y.
   (a) In the jargon, one says that "Y **scales like** X^{β_1}".
   (b) Ignore the noise temporarily, so we have a deterministic relationship $y = y_0 x^{\beta_1}$. A small difference in x, say dx, means a fractional difference of dx/x. The ratio of the fractional difference in y, dy/y to the fractional difference in x, sometimes called the **elasticity** of y with respect to x, is $(dy/y)/(dx/x) = (dy/dx)/(y/x)$, is[2]

$$\frac{\frac{dy}{dx}}{\frac{y}{x}} = \frac{y_0 \beta_1 x^{\beta_1 - 1}}{\frac{y_0 x^{\beta_1}}{x}} = \beta_1 \tag{9.24}$$

[2]If you are appalled by expressions like dy/y, you have good mathematical taste, and are invited to reach this conclusion through a proper proof using limits. (It can be done.)

3. $\mathbb{E}[Y|X=x] \neq y_0 x^{\beta_1}$.
   (a) $y_0 x^{\beta_1}$ is the conditional median, however.
   (b) By parallel reasoning to what we went through above with the log-normal,

$$\begin{aligned}\mathbb{E}[Y|X=x] &= y_0 x^{\beta_1} e^{\sigma^2/2} && (9.25)\\ \mathrm{Var}[Y|X=x] &= y_0^2 x^{2\beta_1} e^{\sigma^2}(e^{\sigma^2}-1) && (9.26)\end{aligned}$$

Notice that this is a *different* statistical model from

$$Y = y_0 X^{\beta_1} + \epsilon \qquad (9.27)$$

which would lead to $\mathbb{E}[Y|X=x] = y_0 X^{\beta_1}$, $\mathrm{Var}[Y|X=x] = \sigma^2$. (It is, unfortunately, often unclear whether people mean a power law with multiplicative, log-normal noise or with additive, normal noise.) I will leave it as an exercise to check how the interpretations change.

Chapter 10

F-Tests, R^2, and Other Distractions

10.1 The F Test

The F distribution with a, b degrees of freedom is *defined* to be the distribution of the ratio

$$\frac{\chi^2_a/a}{\chi^2_b/b}$$

when χ^2_a and χ^2_b are independent.

Since χ^2 distributions arise from sums of Gaussians, F-distributed random variables tend to arise when we are dealing with ratios of sums of Gaussians. The outstanding examples of this are ratios of variances.

10.1.1 F test of $\beta_1 = 0$ vs. $\beta_1 \neq 0$

Let's consider testing the null hypothesis $\beta_1 = 0$ against the alternative $\beta_1 \neq 0$, in the context of the Gaussian-noise simple linear regression model. That is, we won't question, in our mathematics, whether or not the assumptions of that model hold, we'll presume that they all do, and just ask how we can tell whether $\beta_1 = 0$.

We have said, *ad nauseam*, that under the unrestricted model,

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i))^2$$

with

$$\frac{n\hat{\sigma}^2}{\sigma^2} \sim \chi^2_{n-2}$$

This is true no matter what β_1 is, so, in particular, it continues to hold when $\beta_1 = 0$ but we estimate the general model anyway.

The null model is that

$$Y = \beta_0 + \epsilon$$

with $\epsilon \sim N(0, \sigma^2)$, independent of X and independently across measurements. It's an exercise from 36-226 to show (really, remind!) yourself that, in the null model

$$\hat{\beta}_0 = \overline{y} \sim N(\beta_0, \sigma^2/n)$$

It is another exercise to show

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \overline{y})^2 = s_Y^2$$

and

$$\frac{n s_Y^2}{\sigma^2} \sim \chi^2_{n-1}$$

However, s_Y^2 is not independent of $\hat{\sigma}^2$. What *is* statistically independent of $\hat{\sigma}^2$ is the difference

$$s_Y^2 - \hat{\sigma}^2$$

and

$$\frac{n(s_Y^2 - \hat{\sigma}^2)}{\sigma^2} \sim \chi^2_1$$

I will not pretend to give a proper demonstration of this. Rather, to make it plausible, I'll note that $s_Y^2 - \hat{\sigma}^2$ is the extra mean squared error which comes from estimating only one coefficient rather than two, that each coefficient kills one degree of freedom in the data, and the total squared error associated with one degree of freedom, over the entire data set, should be about $\sigma^2 \chi^2_1$.

Taking the previous paragraph on trust, then, let's look at a ratio of variances:

$$\begin{aligned} \frac{s_Y^2 - \hat{\sigma}^2}{\hat{\sigma}^2} &= \frac{n(s_Y^2 - \hat{\sigma}^2)}{n\hat{\sigma}^2} && (10.1) \\ &= \frac{\frac{n(s_Y^2 - \hat{\sigma}^2)}{\sigma^2}}{\frac{n\hat{\sigma}^2}{\sigma^2}} && (10.2) \\ &= \frac{\chi^2_1}{\chi^2_{n-2}} && (10.3) \end{aligned}$$

To get our F distribution, then, we need to use as our test statistic

$$\frac{s_Y^2 - \hat{\sigma}^2}{\hat{\sigma}^2}\frac{n-2}{1} = \left(\frac{s_Y^2}{\hat{\sigma}^2} - 1\right)(n-2)$$

which will have an $F_{1,n-2}$ distribution under the null hypothesis that $\beta_1 = 0$.

Note that the only random, data-dependent part of this is the ratio of $s_Y^2/\hat{\sigma}^2$. We reject the null $\beta_1 = 0$ when this is too large, compared to what's expected under the $F_{1,n-2}$ distribution. Again, this is the distribution of the test statistic *under the null* $\beta_1 = 0$. The variance ratio will tend to be larger under the alternative, with its expected size growing with $|\beta_1|$.

Running this F test in R The easiest way to run the F test for the regression slope on a linear model in R is to invoke the `anova` function, like so:

```
anova(lm(y ~ x))
```

This will give you an analysis-of-variance table for the model. The actual object the function returns is an `anova` object, which is a special type of data frame. The columns record, respectively, degrees of freedom, sums of squares, mean squares, the actual F statistic, and the p value of the F statistic. What we'll care about will be the first row of this table, which will give us the test information for the slope on X.

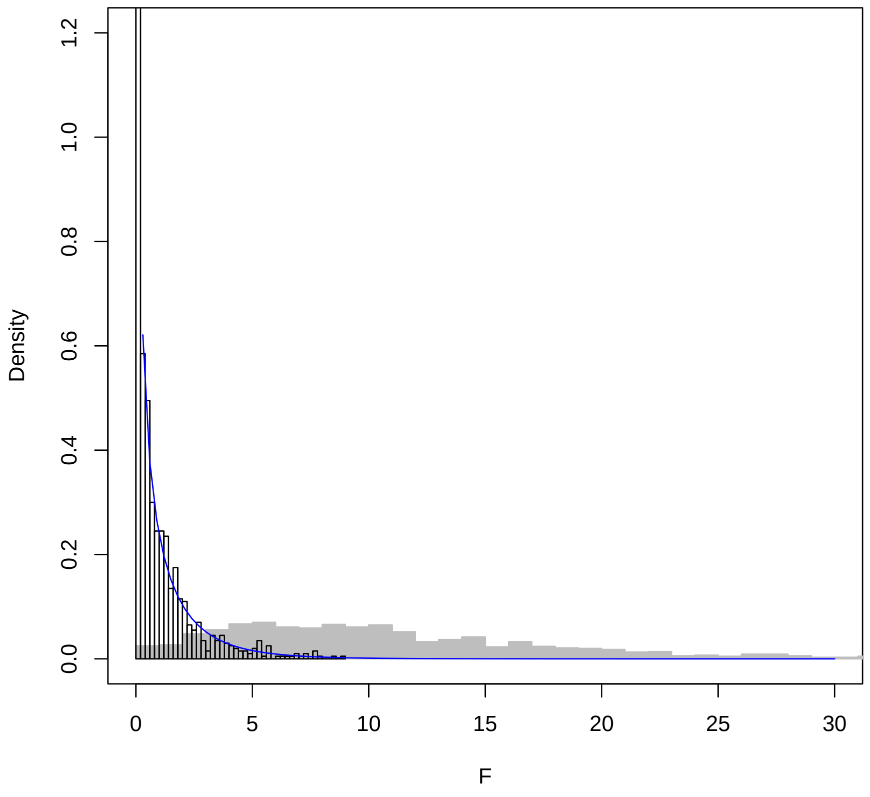
To illustrate more concretely, let's revisit our late friends in Chicago:

```
library(gamair)
data(chicago)
death.temp.lm <- lm(death ~ tmpd, data = chicago)
anova(death.temp.lm)
## Analysis of Variance Table
##
## Response: death
##             Df  Sum Sq Mean Sq F value    Pr(>F)
## tmpd         1  162473  162473  803.07 < 2.2e-16 ***
## Residuals 5112 1034236     202
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As with `summary` on `lm`, the stars are usually a distraction; see Chapter 7 for how to turn them off.

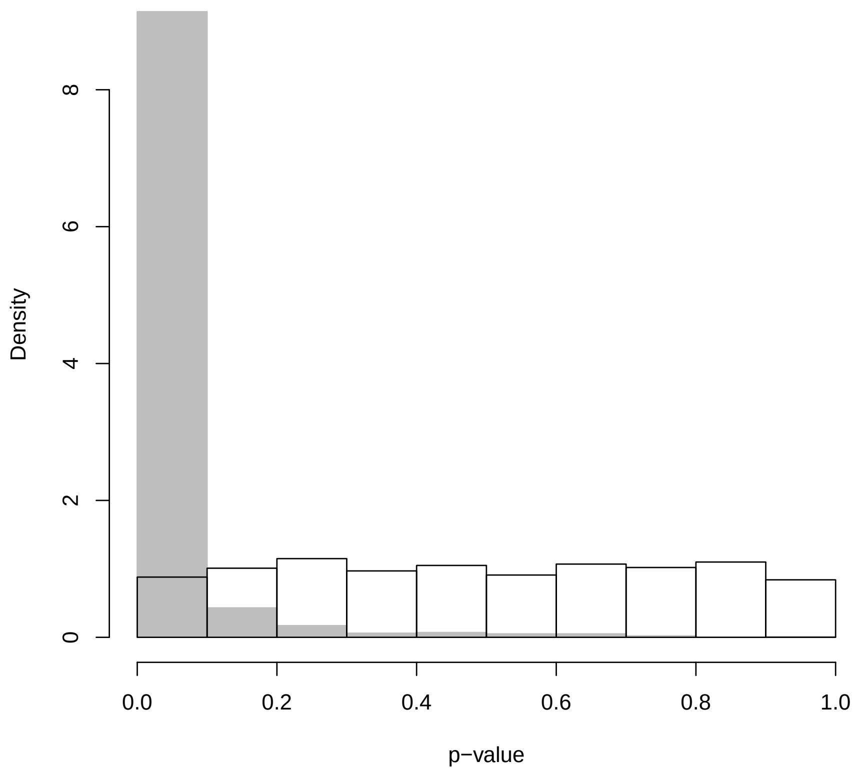
```
# Simulate a Gaussian-noise simple linear regression model Inputs: x sequence;
# intercept; slope; noise variance; switch for whether to return the simulated
# values, or the ratio of s^2_Y/\hat{\sigma}^2 Output: data frame or
# coefficient vector
sim.gnslrm <- function(x, intercept, slope, sigma.sq, var.ratio = TRUE) {
    n <- length(x)
    y <- intercept + slope * x + rnorm(n, mean = 0, sd = sqrt(sigma.sq))
    if (var.ratio) {
        mdl <- lm(y ~ x)
        hat.sigma.sq <- mean(residuals(mdl)^2)
        # R uses the n-1 denominator in var(), but we need the MLE
        s.sq.y <- var(y) * (n - 1)/n
        return(s.sq.y/hat.sigma.sq)
    } else {
        return(data.frame(x = x, y = y))
    }
}

# Parameters
beta.0 <- 5
beta.1 <- 0  # We are simulating under the null!
sigma.sq <- 0.1
x.seq <- seq(from = -5, to = 5, length.out = 42)
```

FIGURE 10.1: *Code setting up a simulation of a Gaussian-noise simple linear regression model, returning either the actual simulated data frame, or just the variance ratio $s^2_Y/\hat{\sigma}^2$.*



```
# Run a bunch of simulations under the null and get all the F statistics Actual
# F statistic is in the 4th column of the output of anova()
f.stats <- replicate(1000, anova(lm(y ~ x, data = sim.gnslrm(x.seq, beta.0, beta.1,
    sigma.sq, FALSE)))[1, 4])
# Store histogram of the F statistics, but hold off on plotting it
null.hist <- hist(f.stats, breaks = 50, plot = FALSE)
# Run a bunch of simulations under the alternative and get all the F statistics
alt.f <- replicate(1000, anova(lm(y ~ x, data = sim.gnslrm(x.seq, beta.0, -0.05,
    sigma.sq, FALSE)))[1, 4])
# Store a histogram of this, but again hold off on plotting
alt.hist <- hist(alt.f, breaks = 50, plot = FALSE)
# Create an empty plot
plot(0, xlim = c(0, 30), ylim = c(0, 1.2), xlab = "F", ylab = "Density", type = "n")
# Add the histogram of F under the alternative, then under the null
plot(alt.hist, freq = FALSE, add = TRUE, col = "grey", border = "grey")
plot(null.hist, freq = FALSE, add = TRUE)
# Finally, the theoretical F distribution
curve(df(x, 1, length(x.seq) - 2), add = TRUE, col = "blue")
```

FIGURE 10.2: *Comparing the actual distribution of F statistics when we simulate under the null model (black histogram) to the theoretical $F_{1,n-2}$ distribution (blue curve), and to the distribution under the alternative $\beta_1 = -0.05$.*



```
# Take the simulated F statistics and convert to p-values
p.vals <- pf(f.stats, 1, length(x.seq) - 2, lower.tail = FALSE)
alt.p <- pf(alt.f, 1, length(x.seq) - 2, lower.tail = FALSE)
hist(alt.p, col = "grey", freq = FALSE, xlab = "p-value", main = "", border = "grey",
    xlim = c(0, 1))
plot(hist(p.vals, plot = FALSE), add = TRUE, freq = FALSE)
```

FIGURE 10.3: *Distribution of p-values from repeated simulations, under the null hypothesis (black) and the alternative (grey). Notice how the p-values under the null are uniformly distributed, while under the alternative they are bunched up towards small values at the left.*

Assumptions In deriving the F distribution, it is absolutely vital that all of the assumptions of the Gaussian-noise simple linear regression model hold: the true model must be linear, the noise around it must be Gaussian, the noise variance must be constant, the noise must be independent of X and independent across measurements. The *only* hypothesis being tested is whether, maintaining all these assumptions, we must reject the flat model $\beta_1 = 0$ in favor of a line at an angle. In particular, the test never doubts that the right model is a straight line.

The "general linear test" As a preview of coming attractions, we can look at what happens when we compare a linear, Gaussian-noise model with p parameters to a restricted Gaussian-noise linear model model with only q free parameters. Each model gives us an estimate of the noise variance, say $\hat{\sigma}^2_A$ and $\hat{\sigma}^2_0$ (respectively); these are just the mean squared residuals in each model. It will not surprise you to learn that, under the null that the smaller, restricted model is true

$$\frac{n(\hat{\sigma}^2_0 - \hat{\sigma}^2_A)}{\sigma^2} \sim \chi^2_{p-q}$$

while

$$\frac{n\hat{\sigma}^2_A}{\sigma^2} \sim \chi^2_{n-p}$$

The F statistic for testing the restriction of the full model to the sub-model is therefore

$$\frac{\hat{\sigma}^2_0 - \hat{\sigma}^2_A}{\hat{\sigma}^2_A}\frac{n-p}{p-q}$$

and it has an $F_{p-q,n-p}$ distribution.

ANOVA Some readers will notice that I made no use of the ponderous machinery of analysis of variance ("ANOVA") which is often wheeled out in connection with the F test. Despite (or because) of all of its size and complexity, this machinery is really just a historical relic. In serious applied work from the modern (say, post-1985) era, I have never seen any study where filling out an ANOVA table for a regression, etc., was at all important.

There is more to be said for analysis of variance where the observations are divided into discrete, categorical groups, and one wants to know about differences between groups versus variation within a group. In a few chapters, when we see how to handle categorical predictor variables, it will turn out that this useful form of ANOVA is actually a special case of linear regression (§14.3).

10.1.2 The Likelihood Ratio Test

The F test is a special case of a much more general procedure, the **likelihood ratio test**, which works as follows. We start with a general model, where the parameter is a vector $\theta = (\theta_1, \theta_2, \ldots \theta_p)$. We contemplate a restriction, where $\theta = (\theta_1, \theta_2, \ldots \theta_q, 0, \ldots 0)$,

$q < p$. (See below on other possible restrictions.) The restricted sub-model is the null hypothesis, and the full model is the alternative.

Both the restricted model and the full model have maximum likelihood estimators; call these $\hat{\theta}$ and $\hat{\Theta}$, respectively. Let's write L for the log-likelihood function, so $L(\hat{\theta})$ is the maximized log-likelihood under the restricted null model, and $L(\hat{\Theta})$ is the maximized log-likelihood under the unrestricted, alternative, full model. Then

$$\Lambda \equiv L(\hat{\Theta}) - L(\hat{\theta})$$

is the log of the **likelihood ratio** between the models (because $\log a/b = \log a - \log b$). Λ is the test statistic in the likelihood ratio test[1].

Under some "regularity" conditions, which I'll sketch in a moment, there is a simple asymptotic distribution for Λ *under the null hypothesis*. As $n \to \infty$

$$2\Lambda \sim \chi^2_{p-q}$$

Let me first try to give a little intuition, then hand-wave at the math, and then work things through for test $\beta_1 = 0$ vs. $\beta_1 \neq 0$.

The null model is, as I said, a restriction of the alternative model. Any possible parameter value for the null model is also allowed for the alternative. This means the parameter space for the null, say ω, is a strict subset of that for the alternative, $\omega \subset \Omega$. The maximum of the likelihood over the larger space *must* be at least as high as the maximum over the smaller space:

$$L(\hat{\Theta}) = \max_{\theta \in \Omega} L(\theta) \geq \max_{\theta \in \omega} L(\theta) = L(\hat{\theta})$$

Thus, $\Lambda \geq 0$. What's more surprising is that its distribution doesn't change with n (asymptotically), and that depends on the difference in the number of free parameters. Because the MLE is consistent, under the null the estimates of $\theta_{q+1}, \theta_{q+2} \ldots \theta_p$ in $\hat{\Theta}$ all converge to zero, because those parameters *are* zero under the null. In fact, they get closer and closer to zero, but end up making larger and larger contributions to L, because L grows with n. The two effects cancel out, and each free parameter ends up contributing one χ^2_1 term.

Why χ^2? Well, for large n, $\hat{\theta}$ and $\hat{\Theta}$ both have Gaussian distributions around the true θ, and the contributions to the log-likelihood end up depending on the squares of parameter estimates. Since the square of a Gaussian is proportional to a χ^2, it's not surprising that we get something χ^2-ish, though it is nice how everything cancels out. I defer a fuller explanation to the option §10.5.

Sketch of the regularity conditions where the likelihood-ratio test has a χ^2 null First, the MLE must be consistent for both models, and must have a Gaussian distribution around the true parameter (for large n). Second, the restricted model has to "lie in the interior" of the unrestricted, alternative model, and not on the boundary. That is, it must make sense in the alternative model for all the zeroed-out parameters

[1]Some people, being a bit pedantic, call it the log-likelihood-ratio test.

to be either positive or negative. (This would be violated, for instance, if one of the parameters set to zero by the null were a variance.) And that's *mostly* it. Again, see §10.5 for more mathematical details.

Testing $\beta_1 = 0$ What's the log-likelihood at the MLE of the simple linear model? Dredging up the log-likelihood function from Chapter 5,

$$L(\hat{\beta}_0, \hat{\beta}_1, \hat{\sigma}^2) = -\frac{n}{2}\log 2\pi - \frac{n}{2}\log \hat{\sigma}^2 - \frac{1}{2\hat{\sigma}^2}\sum_{i=1}^{n}(y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i))^2 \tag{10.4}$$

But

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i))^2$$

Substituting into Eq. 10.4,

$$L(\hat{\beta}_0, \hat{\beta}_1, \hat{\sigma}^2) = -\frac{n}{2}\log 2\pi - \frac{n}{2}\log \hat{\sigma}^2 - \frac{n\hat{\sigma}^2}{2\hat{\sigma}^2} = -\frac{n}{2}(1 + \log 2\pi) - \frac{n}{2}\log \hat{\sigma}^2$$

So we get a constant which doesn't depend on the parameters at all, and then something proportional to $\log \hat{\sigma}^2$.

The intercept-only model works similarly, only *its* estimate of the intercept is $\overline{y}$, and its noise variance, $\hat{\sigma}^2_0$, is just the sample variance of the y_i:

$$L(\overline{y}, 0, s^2_Y) = -\frac{n}{2}(1 + \log 2\pi) - \frac{n}{2}\log s^2_Y$$

Putting these together,

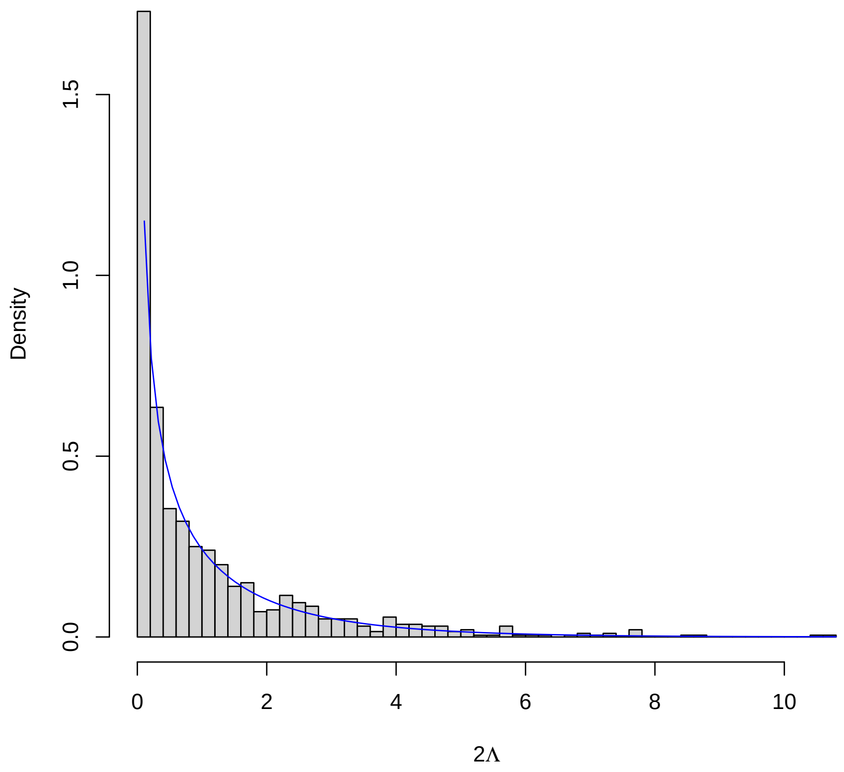
$$L(\hat{\beta}_0, \hat{\beta}_1, \hat{\sigma}^2) - L(\overline{y}, 0, s^2_Y) = \frac{n}{2}\log\frac{s^2_Y}{\hat{\sigma}^2}$$

Thus, under the null hypothesis,

$$n \log \frac{s^2_Y}{\hat{\sigma}^2} \sim \chi^2_1$$

Figure 10.4 shows a simulation confirming this bit of theory.

Connection to *F* tests The ratio $s^2_Y/\hat{\sigma}^2$ is, of course, the F-statistic, up to constants not depending on the data. Since, for this problem, the likelihood ratio test and the F test use equivalent test statistics, if we fix the same size or level α for the two tests, they will have exactly the same power. In fact, even for more complicated linear models — the "general linear tests" — the F test is always equivalent to a likelihood ratio test, at least when the presumptions of the former are met. The likelihood ratio test, however, applies to problems which do not involve Gaussian-noise linear models, while the F test is basically only good for them. If you can only remember *one* of the two tests, remember the likelihood ratio test.



```
# Simulate from the model 1000 times, capturing the variance ratios
var.ratios <- replicate(1000, sim.gnslrm(x.seq, beta.0, beta.1, sigma.sq))
# Convert variance ratios into log likelihood-ratios
LLRs <- (length(x.seq)/2) * log(var.ratios)
# Create a histogram of 2*LLR
hist(2 * LLRs, breaks = 50, freq = FALSE, xlab = expression(2 * Lambda), main = "")
# Add the theoretical chi^2_1 distribution
curve(dchisq(x, df = 1), col = "blue", add = TRUE)
```

FIGURE 10.4: *Comparison of log-likelihood ratios (black histogram) with theoretical χ^2_1 distribution (blue). Note we are simulating under the null hypothesis $\beta_1 = 0$. Can you add a histogram of the distribution under the alternative, and make histograms of p-values, as in Figures 10.2 and 10.3?*

Other constraints Setting $p-q$ parameters to zero is really a special case of imposing $p-q$ linearly independent constraints on the p parameters. For instance, requiring $\theta_2 = \theta_1$ while $\theta_3 = -2\theta_1$ is just as much a two-parameter restriction as fixing $\theta_2 = \theta_3 = 0$. This is because we could transform to a new set of parameters, say $\psi_1 = \theta_1$, $\psi_2 = \theta_2 - \theta_1$, $\psi_3 = \theta_3 + 2\theta_1$, where the restrictions *are* $\psi_2 = \psi_3 = 0$, and we can transform back to the θ parameters without loss of information. So the theory of the likelihood ratio test applies whenever we have linearly independent constraints.

More generally, that theory applies under the following (admittedly rather complicated) conditions:

- Under the null model, θ must obey equations $f_1(\theta) = 0, f_2(\theta) = 0, \ldots f_{p-q}(\theta) = 0$.
- Any θ which obeys those equations is in the null model.
- There is an *invertible* function g where, writing $\psi = g(\theta)$, in the null model, ψ always has $\psi_{q+1}, \ldots \psi_p = 0$, and under the alternative, ψ is unrestricted.

Basically, we need to be able to come up with a change of coordinates where the restrictions amount to fixing some coordinates to zero, but leaving the others alone.

10.2 What the F Test Really Tests

The textbook (§2.7–2.8) goes into great detail about an F test for whether the simple linear regression model "explains" (really, predicts) a "significant" amount of the variance in the response. What this really does is compare two versions of the simple linear regression model. The null hypothesis is that all of the assumptions of that model hold, *and* the slope, β_1, is exactly 0. (This is sometimes called the "intercept-only" model, for obvious reasons.) The alternative is that all of the simple linear regression assumptions hold[2], with $\beta_1 \neq 0$. The alternative, non-zero-slope model will always fit the data better than the null, intercept-only model (why?); the F test asks if the improvement in fit is larger than we'd expect under the null[3].

There are situations where it is useful to know about this precise quantity, and so run an F test on the regression. It is hardly ever, however, a good way to check whether the simple linear regression model is correctly specified, because neither retaining nor rejecting the null gives us information about what we really want to know.

Suppose first that we retain the null hypothesis, i.e., we do not find any significant share of variance associated with the regression. This could be because (i) there is no such variance — the intercept-only model is right; (ii) there is some variance, but we were unlucky; (iii) the test doesn't have enough power to detect departures from the null. To expand on that last point, the power to detect a non-zero slope is going to increase with the sample size n, decrease with the noise level σ^2, and increase with

[2]To get an exact F distribution for the test statistic, we also need the Gaussian-noise assumptions, but under the weaker assumptions of uncorrelated noise, we'll often approach an F distribution asymptotically.

[3]This is also what the likelihood ratio test of §10.1.2 is asking, just with a different notion of measuring fit to the data (likelihood vs. squared error). Everything I'm about to say about F tests applies, suitably modified, to likelihood ratio tests.

the magnitude of the slope $|\beta_1|$. As $\sigma^2/n \rightarrow 0$, the test's power to detect any departures from the null $\rightarrow 1$. If we have a very powerful test, then we can reliably detect departures from the null. If we don't find them, then, we can be pretty sure they're not there. If we have a low-power test, not detecting departures from the null tells us little[4]. If σ^2 is too big or n is too small, our test is inevitably low-powered. Without knowing the power, retaining the null is ambiguous between "there's no signal here" and "we can't tell if there's a signal or not". It would be more useful to look at things like a confidence interval for the regression slope, or, if you must, for σ^2. Of course, there is also possibility (iv), that the real relationship is nonlinear, but the best linear approximation to it has slope (nearly) zero, in which case the F test will have no power to detect the nonlinearity.

Suppose instead that we reject the null, intercept-only hypothesis. *This does not mean that the simple linear model is right.* It means that the latter model predicts better than the intercept-only model — too much better to be due to chance. The simple linear regression model can be absolute garbage, with every single one of its assumptions flagrantly violated, and yet better than the model which makes all those assumptions *and* thinks the optimal slope is zero.

Figure 10.5 provides simulation code for a simple set up where the true regression function is nonlinear and the noise around it has non-constant variance. (Indeed, regression curve is non-monotonic and the noise is multiplicative, not additive.) Still, because a tilted straight line is a much better fit than a flat line, the F test delivers incredibly small p-values — the largest, when I simulate drawing 200 points from the model, is around 10^{-35}, which is about the probability of drawing any particular molecule from 3 billion liters of water. This is the math's way of looking at data like Figure 10.6 and saying "If you want to run a flat line through this, instead of one with a slope, you're crazy"[5]. This is, of course, true; it's just not an answer to "Is simple linear model right here?"

10.2.1 The Essential Thing to Remember

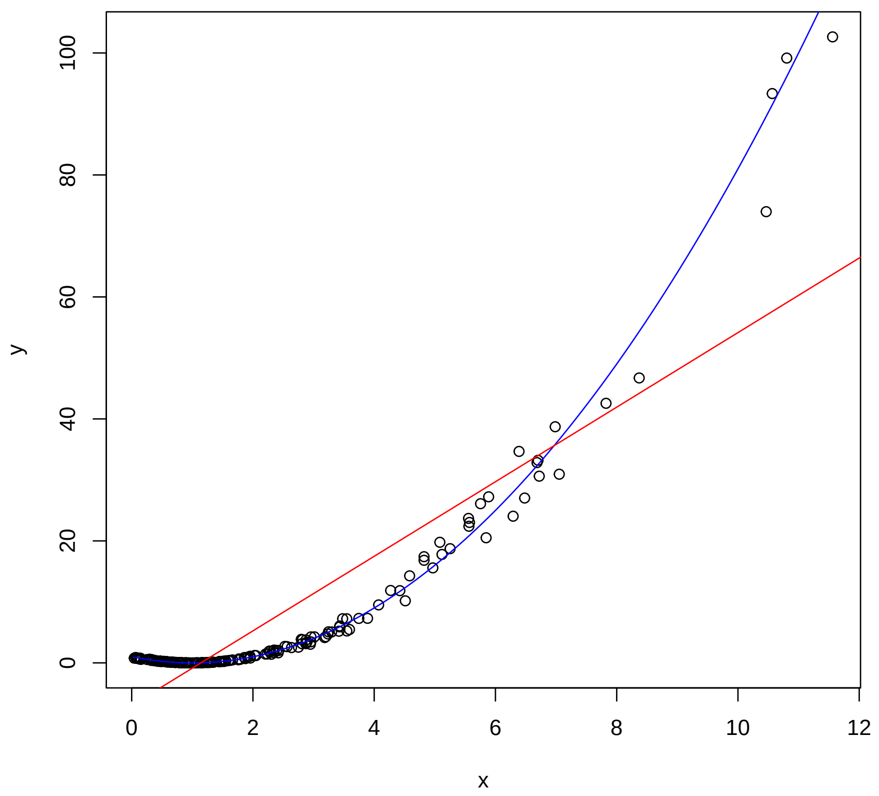
Neither the F test of $\beta_1 = 0$ vs. $\beta_1 \neq 0$ nor the likelihood ratio test nor the Wald/t test of the same hypothesis tell us *anything* about the correctness of the simple linear regression model. All these tests *presume* the simple linear regression model with Gaussian noise is true, and check a special case (flat line) against the general one (titled line). They do not test linearity, constant variance, lack of correlation, or Gaussianity.

[4]Refer back to the discussion of hypothesis testing in Chapter 7.

[5]Similarly, when on p. 10.1.1 we're told the p-value is $\leq 2.2 \times 10^{-16}$, that doesn't mean that there's overwhelming evidence for the simple linear model, it again means that it'd be really stupid to prefer a flat line to a titled one.

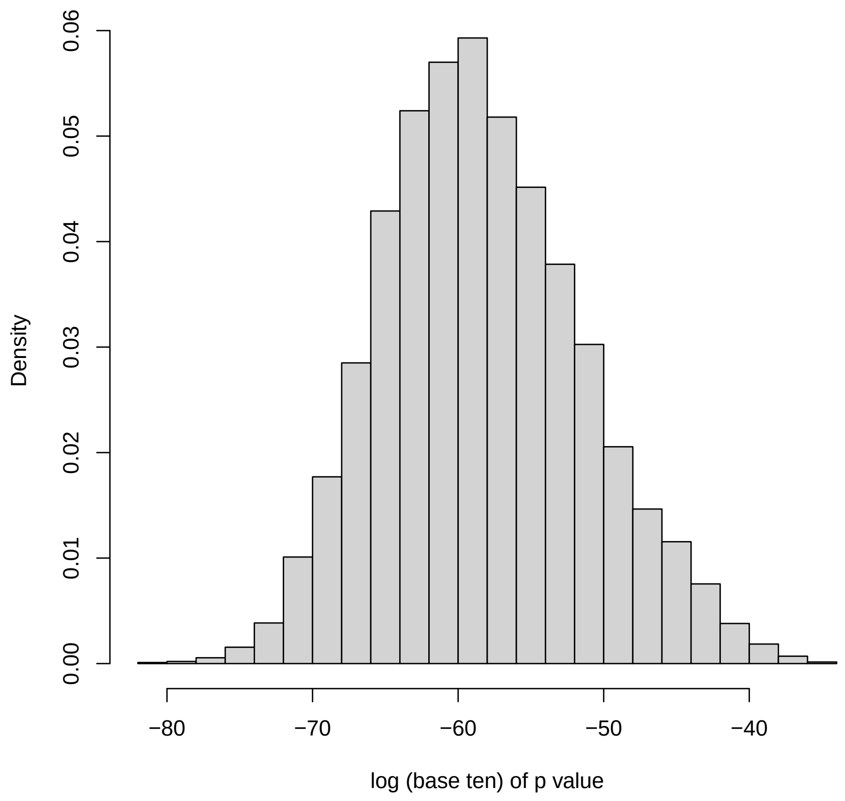
```
# Simulate from a non-linear, non-constant-variance model Curve: Y = (X-1)^2 *
# U U ~ Unif(0.8, 1.2) X ~ Exp(0.5) Inputs: number of data points; whether to
# return data frame or F test of a simple linear model

sim.non.slr <- function(n, do.test = FALSE) {
    x <- rexp(n, rate = 0.5)
    y <- (x - 1)^2 * runif(n, min = 0.8, max = 1.2)
    if (!do.test) {
        return(data.frame(x = x, y = y))
    } else {
        # Fit a linear model, run F test, return p-value
        return(anova(lm(y ~ x))[["Pr(>F)"]][1])
    }
}
```

FIGURE 10.5: *Code to simulate a non-linear model with non-constant variance (in fact, multiplicative rather than additive noise).*



```
not.slr <- sim.non.slr(n = 200)
plot(y ~ x, data = not.slr)
curve((x - 1)^2, col = "blue", add = TRUE)
abline(lm(y ~ x, data = not.slr), col = "red")
```

FIGURE 10.6: *200 points drawn from the non-linear, heteroskedastic model defined in Figure 10.5 (black dots); the true regression curve (blue curve); the least-squares estimate of the simple linear regression (red line). Anyone who's read Chapter 6 and looks at this can realize the linear model is badly wrong here.*



```
f.tests <- replicate(10000, sim.non.slr(n = 200, do.test = TRUE))
hist(log10(f.tests), breaks = 30, freq = FALSE, main = "", xlab = "log (base ten) of p value")
```

FIGURE 10.7: *Distribution of p values from the F test for the simple linear regression model when the data come from the non-linear, heteroskedastic model of Figure 10.5, with sample size of* $n = 200$. *The p-values are all so small that rather than plotting them, I plot their logs in base 10, so the distribution is centered around a p-value of* 10^{-60}*, and the* largest, *least-significant p-values produced in ten thousand simulations were around* 10^{-35}.

10.3 R^2

R^2 has several definitions which are equivalent when we estimate a linear model by least squares. The most basic one is the ratio of the sample variance of the fitted values to the sample variance of Y.

$$R^2 \equiv \frac{s^2_{\hat{m}}}{s^2_Y} \tag{10.5}$$

Alternatively, it's the ratio between the sample covariance of Y and the fitted values, to the sample variance of Y:

$$R^2 = \frac{c_{Y,\hat{m}}}{s^2_Y} \tag{10.6}$$

Let's show that these are equal. Clearly, it's enough to show that the sample variance of $\hat{m}$ equals its covariance with Y. The key observations are that (i) that each $y_i = \hat{m}(x_i) + e_i$, while (ii) the sample covariance between e_i and $\hat{m}(x_i)$ is exactly zero. Thus

$$c_{Y,\hat{m}} = c_{\hat{m}+e,\hat{m}} = s^2_{\hat{m}} + c_{e,\hat{m}} = s^2_{\hat{m}}$$

and we see that, *for linear models estimated by least squares*, Eqs. 10.5 and 10.6 do in fact always give the same result.

That said, what is $s^2_{\hat{m}}$? Since $\hat{m}(x_i) = \hat{\beta}_0 + \hat{\beta}_1 x_i$,

$$s^2_{\hat{m}} = s^2_{\hat{\beta}_0 + \hat{\beta}_1 X} = s^2_{\hat{\beta}_1 X} = \hat{\beta}_1^2 s^2_X$$

We thus get a third expression for R^2:

$$R^2 = \hat{\beta}_1^2 \frac{s^2_X}{s^2_Y} \tag{10.7}$$

From this, we can derive yet a fourth expression:

$$R^2 = \left(\frac{c_{XY}}{s_X s_Y} \right)^2 \tag{10.8}$$

which we can recognize as the squared correlation coefficient between X and Y (hence the square in R^2). A noteworthy feature of this equation is that it shows we get exactly the same R^2 whether we regress Y on X, or regress X on Y.

A final expression for R^2 is

$$R^2 = \frac{s^2_Y - \hat{\sigma}^2}{s^2_Y} \tag{10.9}$$

Since $\hat{\sigma}^2$ is the sample variance of the residuals, and the residuals are uncorrelated (in sample) with $\hat{m}$, it's not hard to show that the numerator is equation to $s^2_{\hat{m}}$.

"Adjusted" R^2 As you remember, $\hat{\sigma}^2$ has a slight negative bias as an estimate of σ^2. One therefore sometimes sees an "adjusted" R^2, using $\frac{n}{n-2}\hat{\sigma}^2$ in place of $\hat{\sigma}^2$, that being an unbiased estimate of σ^2.

Limits for R^2 From Eq. 10.7, it is clear that R^2 will be 0 when $\hat{\beta}_1 = 0$. On the other hand, if all the residuals are zero, then $s_Y^2 = \hat{\beta}^1_2 s_X^2$ and $R^2 = 1$. It is not too hard to show that R^2 can't possible be bigger than 1, so we have marked out the limits: a sample slope of 0 gives an R^2 of 0, the lowest possbile, and all the data points falling exactly on a straight line gives an R^2 of 1, the largest possible.

10.3.1 Theoretical R^2

Suppose we knew the true coefficients. What would R^2 be? Using Eq. 10.5, we'd see

$$R^2 = \frac{\mathrm{Var}[m(X)]}{\mathrm{Var}[Y]} \tag{10.10}$$
$$= \frac{\mathrm{Var}[\beta_0 + \beta_1 X]}{\mathrm{Var}[\beta_0 + \beta_1 X + \epsilon]} \tag{10.11}$$
$$= \frac{\mathrm{Var}[\beta_1 X]}{\mathrm{Var}[\beta_1 X + \epsilon]} \tag{10.12}$$
$$= \frac{\beta_1^2 \mathrm{Var}[X]}{\beta_1^2 \mathrm{Var}[X] + \sigma^2} \tag{10.13}$$

Since all our parameter estimates are consistent, and this formula is continuous in all the parameters, the R^2 we get from our estimate will converge on this limit.

As you will recall from Chapter 1, even if the linear model is totally wrong, our estimate of β_1 will converge on $\mathrm{Cov}[X,Y]/\mathrm{Var}[X]$. Thus, *whether or not the simple linear model applies*, the limiting theoretical R^2 is given by Eq. 10.13, provided we interpret β_1 appropriately.

10.3.2 Distraction or Nuisance?

This section occasioned some discussion on Reddit (link in PDF), which readers may find interesting.

Unfortunately, a lot of myths about R^2 have become endemic in the scientific community, and it is vital at this point to immunize you against them.

1. The most fundamental is that R^2 *does not measure goodness of fit*.

   (a) *R^2 can be arbitrarily low when the model is completely correct.* Look at Eq. 10.13. By making $\mathrm{Var}[X]$ small, or σ^2 large, we drive R^2 towards 0, even when every assumption of the simple linear regression model is correct in every particular.

   (b) *R^2 can be arbitrarily close to 1 when the model is totally wrong.* For a demonstration, the R^2 of the linear model fitted to the simulation in §10.2 is 0.791. There is, indeed, no limit to how high R^2 can get when the true model is nonlinear. All that's needed is for the slope of the best linear approximation to be non-zero, and for $\text{Var}[X]$ to get big.

2. R^2 is also pretty useless as a measure of predictability.
    (a) R^2 *says nothing about prediction error.* Go back to Eq. 10.13, the ideal case: even with σ^2 exactly the same, and no change in the coefficients, R^2 can be anywhere between 0 and 1 just by changing the range of X. Mean squared error is a *much* better measure of how good predictions are; better yet are estimates of out-of-sample error which we'll cover later in the course.
    (b) R^2 *says nothing about interval forecasts.* In particular, it gives us no idea how big prediction intervals, or confidence intervals for $m(x)$, might be.
3. R^2 cannot be compared across data sets: again, look at Eq. 10.13, and see that exactly the same model can have radically different R^2 values on different data.
4. R^2 cannot be compared between a model with untransformed Y and one with transformed Y, or between different transformations of Y. More exactly: it's a free country and no one will stop you from doing that, but it's meaningless; R^2 can easily go down when the model assumptions are better fulfilled, etc.
5. The one situation where R^2 can be compared is when different models are fit to the same data set with the same, untransformed response variable. Then increasing R^2 is the same as decreasing in-sample MSE (by Eq. 10.9). In that case, however, you might as well just compare the MSEs.
6. It is very common to say that R^2 is "the fraction of variance explained" by the regression. This goes along with calling R^2 "the coefficient of determination". These usages arise from Eq. 10.9, and have nothing to recommend them. Eq. 10.8 shows that if we regressed X on Y, we'd get exactly the same R^2. This in itself should be enough to show that a high R^2 says nothing about explaining one variable by another. It is also extremely easy to devise situations where R^2 is high even though neither one could possible explain the other[6]. Unless you want to re-define the verb "to explain" in terms of R^2, there is no connection between it and anything which might be called a scientific explanation[7].

Using adjusted R^2 instead of R^2 does absolutely nothing to fix any of this.

At this point, you might be wondering just what R^2 is good for — what job it does that isn't better done by other tools. The only honest answer I can give you is that I have never found a situation where it helped at all. If I could design the regression curriculum from scratch, I would never mention it. Unfortunately, it lives on as a historical relic, so you need to know what it is, and what mis-understandings about it people suffer from.

[6]Imagine, for example, regressing deaths in Chicago on the number of Chicagoans wearing t-shirts each day. For that matter, imagine regressing the number of Chicagoans wearing t-shirts on the number of deaths. For thousands of examples with even less to recommend them as explanations, see `http://www.tylervigen.com/spurious-correlations`.

[7]Some people (e.g., Weisburd and Piquero 2008; Low-Décarie *et al.* 2014) have attempted to gather all the values of R^2 reported in scientific papers on, say, ecology or crime, to see if ecologists or criminologists have gotten better at explaining the phenomena they study. I hope it's clear why these exercises are pointless.

10.4 The Correlation Coefficient

As you know, the correlation coefficient between X and Y is

$$\rho_{XY} = \frac{\mathrm{Cov}[X,Y]}{\sqrt{\mathrm{Var}[X]\mathrm{Var}[Y]}}$$

which lies between −1 and 1. It takes its extreme values when Y is a linear function of X.

Recall, from Chapter 1, that the slope of the ideal linear predictor β_1 is

$$\frac{\mathrm{Cov}[X,Y]}{\mathrm{Var}[X]}$$

so

$$\rho_{XY} = \beta_1 \sqrt{\frac{\mathrm{Var}[X]}{\mathrm{Var}[Y]}}$$

It's also straightforward to show (Exercise 1) that if we regress $Y/\sqrt{\mathrm{Var}[Y]}$, the *standardized* version of Y, on $X/\sqrt{\mathrm{Var}[X]}$, the standardized version of X, the regression coefficient we'd get would be ρ_{XY}.

In 1954, the great statistician John W. Tukey wrote (Tukey, 1954, p. 721)

> Does anyone know when the correlation coefficient is useful, as opposed to when it is used? If so, why not tell us?

Sixty years later, the world is still waiting for a good answer[8].

10.5 More on the Likelihood Ratio Test

> This section is optional, but *strongly* recommended.

We're assuming that the true parameter value, call it θ, lies in the restricted class of models ω. So there are q components to θ which matter, and the other $p-q$ are fixed by the constraints defining ω. To simplify the book-keeping, let's say those constraints are all that the extra parameters are zero, so $\theta = (\theta_1, \theta_2, \ldots \theta_q, 0, \ldots 0)$, with $p-q$ zeroes at the end.

The restricted MLE $\widehat{\theta}$ obeys the constraints, so

$$\widehat{\theta} = (\widehat{\theta}_1, \widehat{\theta}_2, \ldots \widehat{\theta}_q, 0, \ldots 0) \tag{10.14}$$

The unrestricted MLE does not have to obey the constraints, so it's

$$\widehat{\Theta} = (\widehat{\Theta}_1, \widehat{\Theta}_2, \ldots \widehat{\Theta}_q, \widehat{\Theta}_{q+1}, \ldots \widehat{\Theta}_p) \tag{10.15}$$

[8]To be scrupulously fair, Tukey did admit there was one clear case where correlation coefficients were useful; they are, as we have just seen, basically the slopes in simple linear regressions. But even so, as soon as we have multiple predictors (as we will in two weeks), regression will no longer match up with correlation. Also, *covariances* are useful, but why turn a covariance into a correlation?

Because both MLEs are consistent, we know that $\widehat{\theta}_i \rightarrow \theta_i$, $\widehat{\Theta}_i \rightarrow \theta_i$ if $1 \leq i \leq q$, and that $\widehat{\Theta}_i \rightarrow 0$ if $q+1 \leq i \leq p$.

Very roughly speaking, it's the last extra terms which end up making $L(\widehat{\Theta})$ larger than $L(\widehat{\theta})$. Each of them tends towards a mean-zero Gaussian whose variance is $O(1/n)$, but their impact on the log-likelihood depends on the square of their sizes, and the square of a mean-zero Gaussian has a χ^2 distribution with one degree of freedom. A whole bunch of factors cancel out, leaving us with a sum of $p-q$ independent χ^2_1 variables, which has a χ^2_{p-q} distribution.

In slightly more detail, we know that $L(\widehat{\Theta}) \geq L(\widehat{\theta})$, because the former is a maximum over a larger space than the latter. Let's try to see how big the difference is by doing a Taylor expansion around $\widehat{\Theta}$, which we'll take out to second order.

$$\begin{aligned} L(\widehat{\theta}) &\approx L(\widehat{\Theta}) + \sum_{i=1}^{p}(\widehat{\Theta}_i - \widehat{\theta}_i)\left(\left.\frac{\partial L}{\partial \theta_i}\right|_{\widehat{\Theta}}\right) + \frac{1}{2}\sum_{i=1}^{p}\sum_{j=1}^{p}(\widehat{\Theta}_i - \widehat{\theta}_i)\left(\left.\frac{\partial^2 L}{\partial \theta_i \partial \theta_j}\right|_{\widehat{\Theta}}\right)(\widehat{\Theta}_j - \widehat{\theta}_j) \\ &= L(\widehat{\Theta}) + \frac{1}{2}\sum_{i=1}^{p}\sum_{j=1}^{p}(\widehat{\Theta}_i - \widehat{\theta}_i)\left(\left.\frac{\partial^2 L}{\partial \theta_i \partial \theta_j}\right|_{\widehat{\Theta}}\right)(\widehat{\Theta}_j - \widehat{\theta}_j) \qquad (10.16) \end{aligned}$$

All the first-order terms go away, because $\widehat{\Theta}$ is a maximum of the likelihood, and so the first derivatves are all zero there. Now we're left with the second-order terms. Writing all the partials out repeatedly gets tiresome, so abbreviate $\partial^2 L/\partial \theta_i \partial \theta_j$ as $L_{,ij}$.

To simplify the book-keeping, suppose that the second-derivative matrix, or **Hessian**, is diagonal. (This should seem like a swindle, but we get the same conclusion without this supposition, only we need to use a lot more algebra — we diagonalize the Hessian by an orthogonal transformation.) That is, suppose $L_{,ij} = 0$ unless $i = j$. Now we can write

$$L(\widehat{\Theta}) - L(\widehat{\theta}) \approx -\frac{1}{2}\sum_{i=1}^{p}(\widehat{\Theta}_i - \widehat{\theta}_i)^2 L_{,ii} \qquad (10.17)$$

$$2\left[L(\widehat{\Theta}) - L(\widehat{\theta})\right] \approx -\sum_{i=1}^{q}(\widehat{\Theta}_i - \widehat{\theta}_i)^2 L_{,ii} - \sum_{i=q+1}^{p}(\widehat{\Theta}_i)^2 L_{,ii} \qquad (10.18)$$

At this point, we need a fact about the asymptotic distribution of maximum likelihood estimates: they're generally Gaussian, centered around the true value, and with a shrinking variance that depends on the Hessian evaluated at the true parameter value; this is called the **Fisher information**, F or I. (Call it F.) If the Hessian is diagonal, then we can say that

$$\begin{aligned} \widehat{\Theta}_i &\rightsquigarrow \mathcal{N}(\theta_i, -1/nF_{ii}) \qquad (10.19) \\ \widehat{\theta}_i &\rightsquigarrow \mathcal{N}(\theta_1, -1/nF_{ii})\ 1 \leq i \leq q \qquad (10.20) \\ \widehat{\theta}_i &= 0\ q+1 \leq i \leq p \qquad (10.21) \end{aligned}$$

Also, $(1/n)L_{,ii} \rightarrow -F_{ii}$.

Putting all this together, we see that each term in the second summation in Eq. 10.18 is (to abuse notation a little)

$$\frac{-1}{nF_{ii}}(\mathcal{N}(0,1))^2 L_{,ii} \rightarrow \chi^2_1 \tag{10.22}$$

so the whole second summation has a χ^2_{p-q} distribution. The first summation, meanwhile, goes to zero because $\widehat{\Theta}_i$ and $\widehat{\theta}_i$ are actually strongly correlated, so their difference is $O(1/n)$, and their difference squared is $O(1/n^2)$. Since $L_{,ii}$ is only $O(n)$, that summation drops out.

A somewhat less hand-wavy version of the argument uses the fact that the MLE is really a vector, with a multivariate normal distribution which depends on the inverse of the Fisher information matrix:

$$\widehat{\Theta} \rightsquigarrow \mathcal{N}(\theta, (1/n)F^{-1}) \tag{10.23}$$

Then, at the cost of more linear algebra, we don't have to assume that the Hessian is diagonal.

10.6 Concluding Comment

The tone I have taken when discussing F tests, R^2 and correlation has been dismissive. This is deliberate, because they are grossly abused and over-used in current practice, especially by non-statisticians, and I want you to be too proud (or too ashamed) to engage in those abuses. In a better world, we'd just skip over them, but you will have to deal with colleagues, and bosses, who learned their statistics in the bad old days, and so have to understand what they're doing wrong. ("Science advances funeral by funeral".)

In all fairness, the people who *came up* with these tools were great scientists, struggling with very hard problems when nothing was clear; they were inventing all the tools and concepts we take for granted in a class like this. Anyone in this class, me included, would be doing very well to come up with *one* idea over the whole of our careers which is as good as R^2. But we best respect our ancestors, and the tradition they left us, when we improve that tradition where we can. Sometimes that means throwing out the broken bits.

10.7 Further Reading

Refer back to Chapter 6 on diagnostics for ways of *actually* checking whether the relationship between Y and X is linear (along with the other assumptions of the model). For more on the topic of conducting formal tests of linearity, or other parametric regression specifications, see the chapter "Testing Regression Specifications" in Shalizi (forthcoming).

Refer back to Chapter 7, on parametric inference, for advice on when it is actually interesting to test the hypothesis $\beta_1 = 0$.

Full mathematical treatments of likelihood ratio tests can be found in many textbooks, e.g., Schervish (1995) or Gouriéroux and Monfort (1989/1995, vol. II). The original proof that it has a χ^2_{p-q} asymptotic distribution was given by Wilks (1938). Vuong (1989) provides an interesting and valuable treatment of what happens to the likelihood ratio test when *neither* the null nor the alternative is strictly true, but we want to pick the one which is *closer* to the truth; that paper also develops the theory when the null is not a restriction of the alternative, but rather the two hypotheses come from distinct statistical models.

People have been warning about the fallacy of R^2 to measure goodness of fit for a long time (Anderson and Shanteau, 1977; Birnbaum, 1973), apparently without having much of an impact. (See Hamilton (1996) for a discussion of how academic communities can keep on teaching erroneous ideas long after they've been shown to be wrong, and some speculations about why this happens.)

That R^2 has got nothing to do with *explaining* anything has also been pointed out, time after time, for decades (Berk, 2004). A small demo of just how silly "variance explained" can get, using the Chicago data, can be found at `http://bactra.org/weblog/874.html`. Just what it *does* means to give a proper scientific explanation, and what role statistical models might play in doing so, is a topic full of debate, not to say confusion. Shmueli (2010) attempts to relate some of these debates to the practical conduct of statistical modeling. Personally, I have found Salmon (1984) very helpful in thinking about these issues.

Exercises

1. Define $\tilde{Y} = Y/\sqrt{\mathrm{Var}[Y]}$ and $\tilde{X} = X/\sqrt{\mathrm{Var}[X]}$. Show that the slope of the optimal linear predictor of $\tilde{Y}$ from $\tilde{X}$ is ρ_{XY}.

2. Work through the likelihood ratio test for testing regression through the origin ($\beta_0 = 0$) against the simple linear model ($\beta_0 \neq 0$); that is, write Λ in terms of the sample statistics and simplify as much as possible. Under the null hypothesis, 2Λ follows a χ^2 distribution with a certain number of degrees of freedom: how many?

Chapter 11

Simple Linear Regression in Matrix Format

So far, we have not used any notions, or notation, that goes beyond basic algebra and calculus (and probability). This has forced us to do a fair amount of book-keeping, as it were by hand. This is just about tolerable for the simple linear model, with one predictor variable. It will get intolerable if we have multiple predictor variables. Fortunately, a little application of linear algebra will let us abstract away from a lot of the book-keeping details, and make multiple linear regression hardly more complicated than the simple version[1].

These notes will not remind you of how matrix algebra works. However, they will review some results about *calculus* with matrices, and about expectations and variances with vectors and matrices.

Throughout, bold-faced letters will denote matrices, as $\mathbf{a}$ as opposed to a scalar a.

11.1 Least Squares in Matrix Form

Our data consists of n paired observations of the predictor variable X and the response variable Y, i.e., $(x_1, y_1), \ldots (x_n, y_n)$. We wish to fit the model

$$Y = \beta_0 + \beta_1 X + \epsilon \tag{11.1}$$

where $\mathbb{E}[\epsilon|X = x] = 0$, $\mathrm{Var}[\epsilon|X = x] = \sigma^2$, and ϵ is uncorrelated across measurements[2].

[1]Historically, linear models with multiple predictors evolved before the use of matrix algebra for regression. You may imagine the resulting drudgery.

[2]When I need to also assume that ϵ is Gaussian, and strengthen "uncorrelated" to "independent", I'll say so.

11.1.1 The Basic Matrices

Group all of the observations of the response into a single column ($n \times 1$) matrix $\mathbf{y}$,

$$\mathbf{y} = \left[\begin{array}{c} y_1 \\ y_2 \\ \vdots \\ y_n \end{array} \right] \tag{11.2}$$

Similarly, we group both the coefficients into a single vector (i.e., a 2×1 matrix)

$$\beta = \left[\begin{array}{c} \beta_0 \\ \beta_1 \end{array} \right] \tag{11.3}$$

We'd also like to group the observations of the predictor variable together, but we need something which looks a little unusual at first:

$$\mathbf{x} = \left[\begin{array}{cc} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{array} \right] \tag{11.4}$$

This is an $n \times 2$ matrix, where the first column is always 1, and the second column contains the actual observations of X. We have this apparently redundant first column because of what it does for us when we multiply $\mathbf{x}$ by β:

$$\mathbf{x}\beta = \left[\begin{array}{c} \beta_0 + \beta_1 x_1 \\ \beta_0 + \beta_1 x_2 \\ \vdots \\ \beta_0 + \beta_1 x_n \end{array} \right] \tag{11.5}$$

That is, $\mathbf{x}\beta$ is the $n \times 1$ matrix which contains the point predictions.

The matrix $\mathbf{x}$ is sometimes called the **design matrix**.

11.1.2 Mean Squared Error

At each data point, using the coefficients β results in some error of prediction, so we have n prediction errors. These form a vector:

$$\mathbf{e}(\beta) = \mathbf{y} - \mathbf{x}\beta \tag{11.6}$$

(You can check that this subtracts an $n \times 1$ matrix from an $n \times 1$ matrix.)

When we derived the least squares estimator, we used the mean squared error,

$$MSE(\beta) = \frac{1}{n} \sum_{i=1}^{n} e_i^2(\beta) \tag{11.7}$$

How might we express this in terms of our matrices? I claim that the correct form is

$$MSE(\beta) = \frac{1}{n}\mathbf{e}^T\mathbf{e} \tag{11.8}$$

To see this, look at what the matrix multiplication really involves:

$$[e_1 e_2 \ldots e_n]\left[\begin{array}{c} e_1 \\ e_2 \\ \vdots \\ e_n \end{array}\right] \tag{11.9}$$

This, clearly equals $\sum_i e_i^2$, so the MSE has the claimed form.

Let us expand this a little for further use.

$$\begin{aligned} MSE(\beta) &= \frac{1}{n}\mathbf{e}^T\mathbf{e} && (11.10)\\ &= \frac{1}{n}(\mathbf{y}-\mathbf{x}\beta)^T(\mathbf{y}-\mathbf{x}\beta) && (11.11)\\ &= \frac{1}{n}(\mathbf{y}^T-\beta^T\mathbf{x}^T)(\mathbf{y}-\mathbf{x}\beta) && (11.12)\\ &= \frac{1}{n}\left(\mathbf{y}^T\mathbf{y}-\mathbf{y}^T\mathbf{x}\beta-\beta^T\mathbf{x}^T\mathbf{y}+\beta^T\mathbf{x}^T\mathbf{x}\beta\right) && (11.13) \end{aligned}$$

Notice that $(\mathbf{y}^T\mathbf{x}\beta)^T = \beta^T\mathbf{x}^T\mathbf{y}$. Further notice that this is a 1×1 matrix, so $\mathbf{y}^T\mathbf{x}\beta = \beta^T\mathbf{x}^T\mathbf{y}$. Thus

$$MSE(\beta) = \frac{1}{n}\left(\mathbf{y}^T\mathbf{y}-2\beta^T\mathbf{x}^T\mathbf{y}+\beta^T\mathbf{x}^T\mathbf{x}\beta\right) \tag{11.14}$$

11.1.3 Minimizing the MSE

First, we find the gradient of the MSE with respect to β:

$$\begin{aligned} \nabla MSE(\beta &= \frac{1}{n}\left(\nabla\mathbf{y}^T\mathbf{y}-2\nabla\beta^T\mathbf{x}^T\mathbf{y}+\nabla\beta^T\mathbf{x}^T\mathbf{x}\beta\right) && (11.15)\\ &= \frac{1}{n}\left(0-2\mathbf{x}^T\mathbf{y}+2\mathbf{x}^T\mathbf{x}\beta\right) && (11.16)\\ &= \frac{2}{n}\left(\mathbf{x}^T\mathbf{x}\beta-\mathbf{x}^T\mathbf{y}\right) && (11.17) \end{aligned}$$

We now set this to zero at the optimum, $\widehat{\beta}$:

$$\mathbf{x}^T\mathbf{x}\widehat{\beta}-\mathbf{x}^T\mathbf{y} = 0 \tag{11.18}$$

This equation, for the two-dimensional vector $\widehat{\beta}$, corresponds to our pair of normal or estimating equations for $\hat{\beta}_0$ and $\hat{\beta}_1$. Thus, it, too, is called an estimating equation.

Solving,

$$\widehat{\beta} = (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{y} \tag{11.19}$$

That is, we've got one matrix equation which gives us both coefficient estimates.

If this is right, the equation we've got above should in fact reproduce the least-squares estimates we've already derived, which are of course

$$\hat{\beta}_1 = \frac{c_{XY}}{s^2_X} = \frac{\overline{xy} - \bar{x}\bar{y}}{\overline{x^2} - \bar{x}^2} \tag{11.20}$$

and

$$\hat{\beta}_0 = \overline{y} - \hat{\beta}_1\overline{x} \tag{11.21}$$

Let's see if that's right.

As a first step, let's introduce normalizing factors of $1/n$ into both the matrix products:

$$\widehat{\beta} = (n^{-1}\mathbf{x}^T\mathbf{x})^{-1}(n^{-1}\mathbf{x}^T\mathbf{y}) \tag{11.22}$$

Now let's look at the two factors in parentheses separately, from right to left.

$$\begin{aligned}
\frac{1}{n}\mathbf{x}^T\mathbf{y} &= \frac{1}{n}\left[\begin{array}{cccc} 1 & 1 & \ldots & 1 \\ x_1 & x_2 & \ldots & x_n \end{array}\right]\left[\begin{array}{c} y_1 \\ y_2 \\ \vdots \\ y_n \end{array}\right] & (11.23)\\
&= \frac{1}{n}\left[\begin{array}{c} \sum_i y_i \\ \sum_i x_i y_i \end{array}\right] & (11.24)\\
&= \left[\begin{array}{c} \overline{y} \\ \overline{xy} \end{array}\right] & (11.25)
\end{aligned}$$

Similarly for the other factor:

$$\begin{aligned}
\frac{1}{n}\mathbf{x}^T\mathbf{x} &= \frac{1}{n}\left[\begin{array}{cc} n & \sum_i x_i \\ \sum_i x_i & \sum_i x_i^2 \end{array}\right] & (11.26)\\
&= \left[\begin{array}{cc} 1 & \overline{x} \\ \overline{x} & \overline{x^2} \end{array}\right] & (11.27)
\end{aligned}$$

Now we need to take the inverse:

$$\begin{aligned}
\left(\frac{1}{n}\mathbf{x}^T\mathbf{x}\right)^{-1} &= \frac{1}{\overline{x^2} - \bar{x}^2}\left[\begin{array}{cc} \overline{x^2} & -\overline{x} \\ -\overline{x} & 1 \end{array}\right] & (11.28)\\
&= \frac{1}{s^2_X}\left[\begin{array}{cc} \overline{x^2} & -\overline{x} \\ -\overline{x} & 1 \end{array}\right] & (11.29)
\end{aligned}$$

Let's multiply together the pieces.

$$(\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{y} = \frac{1}{s_X^2}\left[\begin{array}{cc} \overline{x^2} & -\overline{x} \\ -\overline{x} & 1 \end{array}\right]\left[\begin{array}{c} \overline{y} \\ \overline{xy} \end{array}\right] \quad (11.30)$$

$$= \frac{1}{s_X^2}\left[\begin{array}{c} \overline{x^2}\overline{y} - \overline{x}\overline{xy} \\ -\overline{x}\overline{y} + \overline{xy} \end{array}\right] \quad (11.31)$$

$$= \frac{1}{s_X^2}\left[\begin{array}{c} (s_X^2 + \bar{x}^2)\overline{y} - \overline{x}(c_{XY} + \bar{x}\bar{y}) \\ c_{XY} \end{array}\right] \quad (11.32)$$

$$= \frac{1}{s_X^2}\left[\begin{array}{c} s_x^2\overline{y} + \bar{x}^2\overline{y} - \overline{x}c_{XY} - \overline{x}^2\bar{y} \\ c_{XY} \end{array}\right] \quad (11.33)$$

$$= \left[\begin{array}{c} \overline{y} - \frac{c_{XY}}{s_X^2}\overline{x} \\ \frac{c_{XY}}{s_X^2} \end{array}\right] \quad (11.34)$$

which is what it should be.

So: $n^{-1}\mathbf{x}^T\mathbf{y}$ is keeping track of $\overline{y}$ and $\overline{xy}$, and $n^{-1}\mathbf{x}^T\mathbf{x}$ keeps track of $\overline{x}$ and $\overline{x^2}$. The matrix inversion and multiplication then handles all the book-keeping to put these pieces together to get the appropriate (sample) variances, covariance, and intercepts. We don't have to remember that any more; we can just remember the one matrix equation, and then trust the linear algebra to take care of the details.

11.2 Fitted Values and Residuals

Remember that when the coefficient vector is β, the point predictions for each data point are $\mathbf{x}\beta$. Thus the vector of fitted values, $\widehat{\mathbf{m}(\mathbf{x})}$, or $\widehat{\mathbf{m}}$ for short, is

$$\widehat{\mathbf{m}} = \mathbf{x}\widehat{\beta} \quad (11.35)$$

Using our equation for $\widehat{\beta}$,

$$\widehat{\mathbf{m}} = \mathbf{x}(\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{y} \quad (11.36)$$

Notice that the fitted values are linear in $\mathbf{y}$. The matrix

$$\mathbf{H} \equiv \mathbf{x}(\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T \quad (11.37)$$

does not depend on $\mathbf{y}$ at all, but does control the fitted values:

$$\widehat{\mathbf{m}} = \mathbf{H}\mathbf{y} \quad (11.38)$$

If we repeat our experiment (survey, observation...) many times at the same $\mathbf{x}$, we get different $\mathbf{y}$ every time. But $\mathbf{H}$ does not change. The properties of the fitted values are thus largely determined by the properties of $\mathbf{H}$. It thus deserves a name; it's usually called the **hat matrix**, for obvious reasons, or, if we want to sound more respectable, the **influence matrix**.

Let's look at some of the properties of the hat matrix.

1. *Influence* Since $\mathbf{H}$ is not a function of $\mathbf{y}$, we can easily verify that $\partial \widehat{m}_i / \partial y_j = H_{ij}$. Thus, H_{ij} is the rate at which the i^{th} fitted value changes as we vary the j^{th} observation, the "influence" that observation has on that fitted value.
2. *Symmetry* It's easy to see that $\mathbf{H}^T = \mathbf{H}$.
3. *Idempotency* A square matrix $\mathbf{a}$ is called **idempotent**[3] when $\mathbf{a}^2 = \mathbf{a}$ (and so $\mathbf{a}^k = \mathbf{a}$ for any higher power k). Again, by writing out the multiplication, $\mathbf{H}^2 = \mathbf{H}$, so it's idempotent.

Idemopotency, Projection, Geometry Idempotency seems like the most obscure of these properties, but it's actually one of the more important. $\mathbf{y}$ and $\widehat{\mathbf{m}}$ are n-dimensional vectors. If we project a vector $\mathbf{u}$ on to the line in the direction of the length-one vector $\mathbf{v}$, we get

$$\mathbf{v}\mathbf{v}^T\mathbf{u} \tag{11.39}$$

(Check the dimensions: $\mathbf{u}$ and $\mathbf{v}$ are both $n \times 1$, so $\mathbf{v}^T$ is $1 \times n$, and $\mathbf{v}^T\mathbf{u}$ is 1×1.) If we group the first two terms together, like so,

$$(\mathbf{v}\mathbf{v}^T)\mathbf{u} \tag{11.40}$$

where $\mathbf{v}\mathbf{v}^T$ is the $n \times n$ **projection matrix** or **projection operator** for that line. Since $\mathbf{v}$ is a unit vector, $\mathbf{v}^T\mathbf{v} = 1$, and

$$(\mathbf{v}\mathbf{v}^T)(\mathbf{v}\mathbf{v}^T) = \mathbf{v}\mathbf{v}^T \tag{11.41}$$

so the projection operator for the line is idempotent. The geometric meaning of idempotency here is that once we've projected $\mathbf{u}$ on to the line, projecting its image on to the same line doesn't change anything.

Extending this same reasoning, for any linear subspace of the n-dimensional space, there is always some $n \times n$ matrix which projects vectors in arbitrary position down into the subspace, and this projection matrix is always idempotent. It is a bit more convoluted to prove that any idempotent matrix is the projection matrix for some subspace, but that's also true. We will see later how to read off the dimension of the subspace from the properties of its projection matrix.

11.2.1 Residuals

The vector of residuals, $\mathbf{e}$, is just

$$\mathbf{e} \equiv \mathbf{y} - \mathbf{x}\widehat{\beta} \tag{11.42}$$

Using the hat matrix,

$$\mathbf{e} = \mathbf{y} - \mathbf{H}\mathbf{y} = (\mathbf{I} - \mathbf{H})\mathbf{y} \tag{11.43}$$

Here are some properties of $\mathbf{I} - \mathbf{H}$:

[3] From the Latin *idem*, "same", and *potens*, "power".

1. *Influence* $\partial e_i / \partial y_j = (\mathbf{I} - \mathbf{H})_{ij}$.
2. *Symmetry* $(\mathbf{I} - \mathbf{H})^T = \mathbf{I} - \mathbf{H}$.
3. *Idempotency* $(\mathbf{I} - \mathbf{H})^2 = (\mathbf{I} - \mathbf{H})(\mathbf{I} - \mathbf{H}) = \mathbf{I} - \mathbf{H} - \mathbf{H} + \mathbf{H}^2$. But, since $\mathbf{H}$ is idempotent, $\mathbf{H}^2 = \mathbf{H}$, and thus $(\mathbf{I} - \mathbf{H})^2 = (\mathbf{I} - \mathbf{H})$.

Thus,

$$MSE(\widehat{\beta}) = \frac{1}{n}\mathbf{y}^T(\mathbf{I} - \mathbf{H})^T(\mathbf{I} - \mathbf{H})\mathbf{y} \tag{11.44}$$

simplifies to

$$MSE(\widehat{\beta}) = \frac{1}{n}\mathbf{y}^T(\mathbf{I} - \mathbf{H})\mathbf{y} \tag{11.45}$$

11.2.2 Expectations and Covariances

We can of course consider the vector of random variables $\mathbf{Y}$. By our modeling assumptions,

$$\mathbf{Y} = \mathbf{x}\beta + \epsilon \tag{11.46}$$

where ϵ is an $n \times 1$ matrix of random variables, with mean vector $\mathbf{0}$, and variance-covariance matrix $\sigma^2\mathbf{I}$. What can we deduce from this?

First, the expectation of the fitted values:

$$\begin{aligned} \mathbb{E}[\mathbf{HY}] &= \mathbf{H}\mathbb{E}[\mathbf{Y}] && (11.47)\\ &= \mathbf{Hx}\beta + \mathbf{H}\mathbb{E}[\epsilon] && (11.48)\\ &= \mathbf{x}(\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{x}\beta + \mathbf{0} && (11.49)\\ &= \mathbf{x}\beta && (11.50) \end{aligned}$$

which is as it should be, since the fitted values are unbiased.

Next, the variance-covariance of the fitted values:

$$\begin{aligned} \text{Var}[\mathbf{HY}] &= \text{Var}[\mathbf{H}(\mathbf{x}\beta + \epsilon)] && (11.51)\\ &= \text{Var}[\mathbf{H}\epsilon] && (11.52)\\ &= \mathbf{H}\text{Var}[\epsilon]\mathbf{H}^T && (11.53)\\ &= \sigma^2\mathbf{HIH} && (11.54)\\ &= \sigma^2\mathbf{H} && (11.55) \end{aligned}$$

using, again, the symmetry and idempotency of $\mathbf{H}$.

Similarly, the expected residual vector is zero:

$$\mathbb{E}[\mathbf{e}] = (\mathbf{I} - \mathbf{H})(\mathbf{x}\beta + \mathbb{E}[\epsilon]) = \mathbf{x}\beta - \mathbf{x}\beta = \mathbf{0} \tag{11.56}$$

The variance-covariance matrix of the residuals:

$$\begin{aligned}
\text{Var}[\mathbf{e}] &= \text{Var}[(\mathbf{I}-\mathbf{H})(\mathbf{x}\beta+\epsilon)] & (11.57)\\
&= \text{Var}[(\mathbf{I}-\mathbf{H})\epsilon] & (11.58)\\
&= (\mathbf{I}-\mathbf{H})\text{Var}[\epsilon](\mathbf{I}-\mathbf{H}))^T & (11.59)\\
&= \sigma^2(\mathbf{I}-\mathbf{H})(\mathbf{I}-\mathbf{H})^T & (11.60)\\
&= \sigma^2(\mathbf{I}-\mathbf{H}) & (11.61)
\end{aligned}$$

Thus, the variance of each residual is not quite σ^2, nor (unless $\mathbf{H}$ is diagonal) are the residuals exactly uncorrelated with each other.

Finally, the expected MSE is

$$\mathbb{E}\left[\frac{1}{n}\mathbf{e}^T\mathbf{e}\right] \qquad (11.62)$$

which is

$$\frac{1}{n}\mathbb{E}\left[\epsilon^T(\mathbf{I}-\mathbf{H})\epsilon\right] \qquad (11.63)$$

We know (because we proved it in the exam) that this must be $(n-2)\sigma^2/n$; we'll see next time how to show this.

11.3 Sampling Distribution of Estimators

Let's now "turn on" the Gaussian-noise assumption, so the noise terms ϵ_i all have the distribution $N(0,\sigma^2)$, and are independent of each other and of X. The vector of all n noise terms, ϵ, is an $n \times 1$ matrix. Its distribution is a **multivariate Gaussian** or **multivariate normal**[4], with mean vector 0, and variance-covariance matrix $\sigma^2\mathbf{I}$. We may use this to get the sampling distribution of the estimator $\widehat{\beta}$:

$$\begin{aligned}
\widehat{\beta} &= (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{Y} & (11.64)\\
&= (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T(\mathbf{x}\beta+\epsilon) & (11.65)\\
&= (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{x}\beta+(\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\epsilon & (11.66)\\
&= \beta+(\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\epsilon & (11.67)
\end{aligned}$$

Since ϵ is Gaussian and is being multiplied by a non-random matrix, $(\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\epsilon$ is also Gaussian. Its mean vector is

$$\mathbb{E}\left[(\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\epsilon\right]=(\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbb{E}[\epsilon]=0 \qquad (11.68)$$

while its variance matrix is

$$\begin{aligned}
\text{Var}\left[(\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\epsilon\right] &= (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\text{Var}[\epsilon]\left((\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\right)^T & (11.69)\\
&= (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\sigma^2\mathbf{I}\mathbf{x}(\mathbf{x}^T\mathbf{x})^{-1} & (11.70)\\
&= \sigma^2(\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{x}(\mathbf{x}^T\mathbf{x})^{-1} & (11.71)\\
&= \sigma^2(\mathbf{x}^T\mathbf{x})^{-1} & (11.72)
\end{aligned}$$

[4]Some people write this distribution as MVN, and others also as N. I will stick to the former.

Since $\mathrm{Var}\left[\widehat{\beta}\right] = \mathrm{Var}\left[(\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\epsilon\right]$ (why?), we conclude that

$$\widehat{\beta} \sim MVN(\beta, \sigma^2(\mathbf{x}^T\mathbf{x})^{-1}) \tag{11.73}$$

Re-writing slightly,

$$\widehat{\beta} \sim MVN(\beta, \frac{\sigma^2}{n}(n^{-1}\mathbf{x}^T\mathbf{x})^{-1}) \tag{11.74}$$

will make it easier to prove to yourself that, according to this, $\widehat{\beta}_0$ and $\widehat{\beta}_1$ are both unbiased (which we know is right), that $\mathrm{Var}\left[\hat{\beta}_1\right] = \frac{\sigma^2}{n}s_X^2$ (which we know is right) and that $\mathrm{Var}\left[\hat{\beta}_0\right] = \frac{\sigma^2}{n}(1 + \bar{x}^2/s_X^2)$ (which we know is right). This will also give us $\mathrm{Cov}\left[b\hat{e}ta_0, \hat{\beta}_1\right]$, which otherwise would be tedious to calculate.

I will leave you to show, in a similar way, that the fitted values $\mathbf{Hy}$ are multivariate Gaussian, as are the residuals $\mathbf{e}$, and to find both their mean vectors and their variance matrices.

11.4 Derivatives with Respect to Vectors

This is a brief review, not intended as a full course in vector calculus.

Consider some scalar function of a vector, say $f(\mathbf{x})$, where $\mathbf{x}$ is represented as a $p \times 1$ matrix. (Here $\mathbf{x}$ is just being used as a place-holder or generic variable; it's not necessarily the design matrix of a regression.) We would like to think about the derivatives of f with respect to $\mathbf{x}$.

Derivatives are rates of change; they tell us how rapidly the function changes in response to minute changes in its arguments. Since $\mathbf{x}$ is a $p \times 1$ matrix, we could also write

$$f(\mathbf{x}) = f(x_1, x_1, x_p) \tag{11.75}$$

This makes it clear that f will have a partial derivative with respect to each component of $\mathbf{x}$. How much does f change when we vary the components? We can find this out by using a Taylor expansion. If we pick some base value of the matrix $\mathbf{x}^0$ and expand around it,

$$\begin{aligned} f(\mathbf{x}) &\approx f(\mathbf{x}^0) + \sum_{i=1}^{p}(\mathbf{x}-\mathbf{x}^0)_i \left.\frac{\partial f}{\partial x_i}\right|_{\mathbf{x}^0} && (11.76) \\ &= f(\mathbf{x}^0) + (\mathbf{x}-\mathbf{x}^0)^T \nabla f(\mathbf{x}^0) && (11.77) \end{aligned}$$

where we *define* the gradient, ∇f, to be the vector of partial derivatives,

$$\nabla f \equiv \left[\begin{array}{c} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_p} \end{array}\right] \tag{11.78}$$

Notice that this defines ∇f to be a one-column matrix, just as $\mathbf{x}$ was taken to be. You may sometimes encounter people who want it to be a one-*row* matrix; it comes to the same thing, but you may have to track a lot of transposes to make use of their math.

All of the properties of the gradient can be proved using those of partial derivatives. Here are some basic ones we'll need.

1. *Linearity*
$$\nabla (af(\mathbf{x}) + bg(\mathbf{x})) = a\nabla f(\mathbf{x}) + b\nabla g(\mathbf{x}) \tag{11.79}$$
PROOF: Directly from the linearity of partial derivatives.

2. *Linear forms* If $f(\mathbf{x}) = \mathbf{x}^T\mathbf{a}$, with $\mathbf{a}$ not a function of $\mathbf{x}$, then
$$\nabla(\mathbf{x}^T\mathbf{a}) = \mathbf{a} \tag{11.80}$$
PROOF: $f(\mathbf{x}) = \sum_i x_i a_i$, so $\partial f/\partial x_i = a_i$. Notice that $\mathbf{a}$ was already a $p \times 1$ matrix, so we don't have to transpose anything to get the derivative.

3. *Linear forms the other way* If $f(\mathbf{x}) = \mathbf{b}\mathbf{x}$, with $\mathbf{b}$ not a function of $\mathbf{x}$, then
$$\nabla(\mathbf{b}\mathbf{x}) = \mathbf{b}^T \tag{11.81}$$
PROOF: Once again, $\partial f/\partial x_i = b_i$, but now remember that $\mathbf{b}$ was a $1 \times p$ matrix, and ∇f is $p \times 1$, so we need to transpose.

4. *Quadratic forms* Let $\mathbf{c}$ be a $p \times p$ matrix which is not a function of $\mathbf{x}$, and consider the **quadratic form** $\mathbf{x}^T\mathbf{c}\mathbf{x}$. (You can check that this is scalar.) The gradient is
$$\nabla(\mathbf{x}^T\mathbf{c}\mathbf{x}) = (\mathbf{c} + \mathbf{c}^T)\mathbf{x} \tag{11.82}$$
PROOF: First, write out the matrix multiplications as explicit sums:
$$\mathbf{x}^T\mathbf{c}\mathbf{x} = \sum_{j=1}^{p} x_j \sum_{k=1}^{p} c_{jk}x_k = \sum_{j=1}^{p}\sum_{k=1}^{p} x_j c_{jk} x_k \tag{11.83}$$
Now take the derivative with respect to x_i.
$$\frac{\partial f}{\partial x_i} = \sum_{j=1}^{p}\sum_{k=1}^{p} \frac{\partial x_j c_{jk} x_k}{\partial x_i} \tag{11.84}$$
If $j = k = i$, the term in the inner sum is $2c_{ii}x_i$. If $j = i$ but $k \neq i$, the term in the inner sum is $c_{ik}x_k$. If $j \neq i$ but $k = i$, we get $x_j c_{ji}$. Finally, if $j \neq i$ and $k \neq i$, we get zero. The $j = i$ terms add up to $(\mathbf{c}\mathbf{x})_i$. The $k = i$ terms add up to $(\mathbf{c}^T\mathbf{x})_i$. (This splits the $2c_{ii}x_i$ evenly between them.) Thus
$$\frac{\partial f}{\partial x_i} = ((\mathbf{c} + \mathbf{c}^T\mathbf{x})_i \tag{11.85}$$
and
$$\nabla f = (\mathbf{c} + \mathbf{c}^T)\mathbf{x} \tag{11.86}$$
(You can, and should, double check that this has the right dimensions.)

5. *Symmetric quadratic forms* If $\mathbf{c} = \mathbf{c}^T$, then

$$\nabla \mathbf{x}^T \mathbf{c}\mathbf{x} = 2\mathbf{c}\mathbf{x} \tag{11.87}$$

11.4.1 Second Derivatives

The $p \times p$ matrix of second partial derivatives is called the **Hessian**. I won't step through its properties, except to note that they, too, follow from the basic rules for partial derivatives.

11.4.2 Maxima and Minima

We need all the partial derivatives to be equal to zero at a minimum or maximum. This means that the gradient must be zero there. At a minimum, the Hessian must be positive-definite (so that moves away from the minimum always increase the function); at a maximum, the Hessian must be negative definite (so moves away always decrease the function). If the Hessian is neither positive nor negative definite, the point is neither a minimum nor a maximum, but a "saddle" (since moving in some directions increases the function but moving in others decreases it, as though one were at the center of a horse's saddle).

11.5 Expectations and Variances with Vectors and Matrices

If we have p random variables, $Z_1, Z_2, \ldots Z_p$, we can grow them into a random vector $\mathbf{Z} = [Z_1 Z_2 \ldots Z_p]^T$. (That is, the random vector is an $n \times 1$ matrix of random variables.)

This has an expected value:

$$\mathbb{E}[Z] \equiv \int \mathbf{z} p(\mathbf{z}) d\mathbf{z} \tag{11.88}$$

and a little thought shows

$$\mathbb{E}[Z] = \left[\begin{array}{c} \mathbb{E}[Z_1] \\ \mathbb{E}[Z_2] \\ \vdots \\ \mathbb{E}\left[Z_p\right] \end{array} \right] \tag{11.89}$$

Since expectations of random scalars are linear, so are expectations of random vectors: when a and b are non-random scalars,

$$\mathbb{E}[a\mathbf{Z} + b\mathbf{W}] = a\mathbb{E}[\mathbf{Z}] + b\mathbb{E}[\mathbf{W}] \tag{11.90}$$

If $\mathbf{a}$ is a non-random matrix,

$$\mathbb{E}[\mathbf{a}\mathbf{Z}] = \mathbf{a}\mathbb{E}[\mathbf{Z}] \tag{11.91}$$

Every coordinate of a random vector has some covariance with every other coordinate. The variance-covariance matrix of $\mathbf{Z}$ is the $p \times p$ matrix which stores these:

$$\text{Var}[\mathbf{Z}] \equiv \left[\begin{array}{cccc} \text{Var}[Z_1] & \text{Cov}[Z_1, Z_2] & \ldots & \text{Cov}\left[Z_1, Z_p\right] \\ \text{Cov}[Z_2, Z_1] & \text{Var}[Z_2] & \ldots & \text{Cov}\left[Z_2, Z_p\right] \\ \vdots & \vdots & \vdots & \vdots \\ \text{Cov}\left[Z_p, Z_1\right] & \text{Cov}\left[Z_p, Z_2\right] & \ldots & \text{Var}\left[Z_p\right] \end{array}\right] \tag{11.92}$$

This inherits properties of ordinary variances and covariances. Just $\text{Var}[Z] = \mathbb{E}\left[Z^2\right] - (\mathbb{E}[Z])^2$,

$$\text{Var}[\mathbf{Z}] = \mathbb{E}\left[\mathbf{Z}\mathbf{Z}^T\right] - \mathbb{E}[\mathbf{Z}](\mathbb{E}[\mathbf{Z}])^T \tag{11.93}$$

For a non-random vector $\mathbf{a}$ and a non-random scalar b,

$$\text{Var}[\mathbf{a} + b\mathbf{Z}] = b^2 \text{Var}[\mathbf{Z}] \tag{11.94}$$

For a non-random matrix $\mathbf{c}$,

$$\text{Var}[\mathbf{c}\mathbf{Z}] = \mathbf{c}\text{Var}[\mathbf{Z}]\mathbf{c}^T \tag{11.95}$$

(Check that the dimensions all conform here: if $\mathbf{c}$ is $q \times p$, $\text{Var}[\mathbf{c}\mathbf{Z}]$ should be $q \times q$, and so is the right-hand side.)

For a quadratic form, $\mathbf{Z}^T\mathbf{c}\mathbf{Z}$, with non-random $\mathbf{c}$, the expectation value is

$$\mathbb{E}\left[\mathbf{Z}^T\mathbf{c}\mathbf{Z}\right] = \mathbb{E}[\mathbf{Z}]^T \mathbf{c}\mathbb{E}[\mathbf{Z}] + \text{tr}\,\mathbf{c}\text{Var}[\mathbf{Z}] \tag{11.96}$$

where tr is of course the trace of a matrix, the sum of its diagonal entries. To see this, notice that

$$\mathbf{Z}^T\mathbf{c}\mathbf{Z} = \text{tr}\,\mathbf{Z}^T\mathbf{c}\mathbf{Z} \tag{11.97}$$

because it's a 1×1 matrix. But the trace of a matrix product doesn't change when we cyclic permute the matrices, so

$$\mathbf{Z}^T\mathbf{c}\mathbf{Z} = \text{tr}\,\mathbf{c}\mathbf{Z}\mathbf{Z}^T \tag{11.98}$$

Therefore

$$\begin{aligned} \mathbb{E}\left[\mathbf{Z}^T\mathbf{c}\mathbf{Z}\right] &= \mathbb{E}\left[\text{tr}\,\mathbf{c}\mathbf{Z}\mathbf{Z}^T\right] && (11.99) \\ &= \text{tr}\,\mathbb{E}\left[\mathbf{c}\mathbf{Z}\mathbf{Z}^T\right] && (11.100) \\ &= \text{tr}\,\mathbf{c}\mathbb{E}\left[\mathbf{Z}\mathbf{Z}^T\right] && (11.101) \\ &= \text{tr}\,\mathbf{c}(\text{Var}[\mathbf{Z}] + \mathbb{E}[\mathbf{Z}]\mathbb{E}[\mathbf{Z}]^T) && (11.102) \\ &= \text{tr}\,\mathbf{c}\text{Var}[\mathbf{Z}] + \text{tr}\,\mathbf{c}\mathbb{E}[\mathbf{Z}]\mathbb{E}[\mathbf{Z}]^T) && (11.103) \\ &= \text{tr}\,\mathbf{c}\text{Var}[\mathbf{Z}] + \text{tr}\,\mathbb{E}[\mathbf{Z}]^T \mathbf{c}\mathbb{E}[\mathbf{Z}] && (11.104) \\ &= \text{tr}\,\mathbf{c}\text{Var}[\mathbf{Z}] + \mathbb{E}[\mathbf{Z}]^T \mathbf{c}\mathbb{E}[\mathbf{Z}] && (11.105) \end{aligned}$$

using the fact that tr is a linear operation so it commutes with taking expectations; the decomposition of $\text{Var}[\mathbf{Z}]$; the cyclic permutation trick again; and finally dropping tr from a scalar.

Unfortunately, there is generally no simple formula for the variance of a quadratic form, unless the random vector is Gaussian.

11.6 Further Reading

Linear algebra is a pre-requisite for this class; I strongly urge you to go back to your textbook and notes for review, if any of this is rusty. If you desire additional resources, I recommend Axler (1996) as a concise but thorough course. Petersen and Pedersen (2012), while not an introduction or even really a review, is an extremely handy compendium of matrix, and matrix-calculus, identities.

Part II

Regression with Multiple Predictors

Chapter 12

Multiple Linear Regression

12.1 Recap on Simple Linear Regression in Matrix Form

Let's start with a brief summary of re-doing simple linear regression with matrices. We collect all our observations of the response variable into a vector, which we write as an $n \times 1$ matrix $\mathbf{y}$, one row per data point. We group the two coefficients into a 2×1 matrix β. We create an $n \times 2$ matrix $\mathbf{x}$, where the first column consists entirely of 1s, and the second column contains all our observations of the predictor variable, again, one row per data point. Our point predictions are then given by $\mathbf{x}\beta$, and the mean squared error by $n^{-1}(\mathbf{y}-\mathbf{x}\beta)^T(\mathbf{y}-\mathbf{x}\beta)$.

The derivative of the MSE with respect to β is

$$\frac{2}{n}(-\mathbf{x}^T\mathbf{y}+\mathbf{x}^T\mathbf{x}\beta) \tag{12.1}$$

Setting this to zero at the optimum coefficient vector $\widehat{\beta}$ gives the (matrix) estimating equation

$$-\mathbf{x}^T\mathbf{y}+\mathbf{x}^T\mathbf{x}\widehat{\beta} = 0 \tag{12.2}$$

whose solution is of course

$$\widehat{\beta} = (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{y} \tag{12.3}$$

We verified last time that $\widehat{\beta}$ does, in fact, coincide with what we already know the least squares solutions to be. Before, we had two estimating equations for two unknowns ($\hat{\beta}_0$ and $\hat{\beta}_1$), and we had to keep track of how they related to each other and how to solve either one. The matrix inversion and multiplication in Eq. 12.3 encapsulates all of that book-keeping.

We also saw that the fitted values at the data points used to estimate the model are linear combinations of the observed responses, with weights given by the **hat** or **influence** matrix:

$$\widehat{\mathbf{m}} = \mathbf{x}(\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{y} = \mathbf{H}\mathbf{y} \tag{12.4}$$

Geometrically, this means that we find the fitted values by taking the vector of observed responses **y** and projecting it on to a certain plane, which is entirely defined by the values in **x**.

12.2 Multiple Linear Regression

We are now ready to go from the *simple* linear regression model, with one predictor variable, to em multiple linear regression models, with more than one predictor variable[1]. Let's start by presenting the statistical model, and get to estimating it in just a moment.

12.2.1 The Statistical Model, without Assuming Gaussian Noise

In the basic form of the **multiple linear regression model**,

1. There are p quantitative predictor variables, $X_1, X_2, \ldots X_p$. We make no assumptions about their distribution; in particular, they may or may not be dependent. X without a subscript will refer to the vector of all of these taken together.
2. There is a single response variable Y.
3. $Y = \beta_0 + \sum_{i=1}^{p} \beta_i X_i + \epsilon$, for some constants (coefficients) $\beta_0, \beta_1, \ldots \beta_p$.
4. The noise variable ϵ has $\mathbb{E}[\epsilon | X = x] = 0$ (mean zero), $\mathrm{Var}[\epsilon | X = x] = \sigma^2$ (constant variance), and is uncorrelated across observations.

In matrix form, when we have n observations,

$$\mathbf{Y} = \mathbf{X}\beta + \epsilon \tag{12.5}$$

where $\mathbf{X}$ is a $n \times (p+1)$ matrix of random variables (including an all-and-always 1 first column), and ϵ is an $n \times 1$ matrix of noise variables. By the modeling assumptions, $\mathbb{E}[\epsilon | \mathbf{X}] = 0$ while $\mathrm{Var}[\epsilon | \mathbf{X}] = \sigma^2 \mathbf{I}$.

12.2.2 The Statistical Model, Assuming Gaussian Noise

In the **multiple linear regression model with Gaussian noise**,

1. There are p quantitative predictor variables, $X_1, X_2, \ldots X_p$. We make no assumptions about their distribution; in particular, they may or may not be dependent. X without a subscript will refer to the vector of all of these taken together.
2. There is a single response variable Y.

[1]You might wonder why the jargon here contrasts "simple" with "multiple", rather than with "complex". The reason is that the older sense of "simple" is "having only one part" or "made from just one ingredient".

3. The variables are related through

$$Y = \beta_0 + \sum_{i=1}^{p} \beta_i X_i + \epsilon \ , \tag{12.6}$$

   for some constants (coefficients) $\beta_0, \beta_1, \ldots \beta_p$.

4. The noise variables ϵ have a jointly-Gaussian $MVN(0, \sigma^2 \mathbf{I})$ distribution, independent of $\mathbf{X}$.

From these assumptions, it follows that, conditional on $\mathbf{X}$, $\mathbf{Y}$ has a multivariate Gaussian distribution,

$$\mathbf{Y}|\mathbf{X} \sim MVN(\mathbf{X}\beta, \sigma^2 \mathbf{I}) \tag{12.7}$$

12.2.3 Parameter Interpretation

β_0 is the expected value of Y are the origin:

$$\beta_0 = \mathbb{E}\left[Y|X_1 = 0, X_2 = 0, \ldots X_p = 0\right] \tag{12.8}$$

The multiple linear regression model assumes that each predictor variable makes a separate contribution to the expected response, that these contributions add up without any interaction, and that each predictor's contribution is linear[2]. Thus β_i is the rate at which $\mathbb{E}[Y]$ changes as X_i, and *only* X_i, changes, regardless of where X_i starts (linearity in X_i), and regardless of what any of the other variables might be (additivity across variables).

12.3 Derivation of the Least Squares Estimator

We now wish to estimate the model by least squares. Fortunately, we did essentially all of the necessary work last time.

This is because the formula we derived for the mean squared error,

$$\frac{1}{n}(\mathbf{y} - \mathbf{x}\beta)^T(\mathbf{y} - \mathbf{x}\beta) \tag{12.9}$$

did not actually care whether $\mathbf{x}$ was $n \times 2$ or $n \times (p+1)$ for any larger p, so long as β was $(p+1) \times 1$. Neither did any of the matrix calculus we did, so it remains true that

$$\nabla_\beta MSE(\beta) = \frac{2}{n}\left(-\mathbf{x}^T\mathbf{y} + \mathbf{x}^T\mathbf{x}\beta\right) \ ; \tag{12.10}$$

that the estimating equation is

$$-\mathbf{x}^T\mathbf{y} + \mathbf{x}^T\mathbf{x}\widehat{\beta} = 0 \tag{12.11}$$

[2]We will see some ways of allowing predictor variables to interact later in this class. The topic will be explored more fully in 402, along with additive but nonlinear models.

and that the solution, the **ordinary least squares** (OLS) estimator, is

$$\widehat{\beta} = (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{y} \tag{12.12}$$

Eq. 20.18 is going to keep coming up again and again; whether you memorize it deliberately or through sheer exposure is up to you.

(We didn't *have* to use matrix notation to arrive at this point. In principle, we could have written out the MSE as an explicit sum over data points, and then taken $p+1$ partial derivatives with respect to the $p+1$ coefficients. This would have led to a system of $p+1$ linear equations in $p+1$ unknowns, which we could then try to solve. But all of this machinery is conveniently assembled into the linear algebra, which makes it *much* easier to handle.)

12.3.1 Slightly Alternate Derivation

To appreciate what's going on in Eq. 20.18, it may help to look at a *slightly* different derivation, which explicitly separates the intercept from the other coefficients. So, in this subsection, *and this sub-section only*, β_0 will be the scalar intercept, β will be a $p \times 1$ vector of slope coefficients (not $(p+1) \times 1$!), and $\mathbf{x}$ will be an $n \times p$ matrix of observations of the predictors (i.e., no column of 1s).

The mean squared error will be

$$\frac{1}{n}(\mathbf{y} - \beta_0\mathbf{1} - \mathbf{x}\beta)^T(\mathbf{y} - \beta_0\mathbf{1} - \mathbf{x}\beta) \tag{12.13}$$

where $\mathbf{1}$ is the $n \times 1$ matrix of all 1s. The relevant derivatives are

$$\frac{\partial MSE}{\partial \beta_0} = -\frac{2}{n}\mathbf{1}^T(\mathbf{y} - \beta_0\mathbf{1} - \mathbf{x}\beta) \tag{12.14}$$

and

$$\nabla_{\beta} MSE = \frac{2}{n}(\beta_0\mathbf{x}^T\mathbf{1} - \mathbf{x}^T\mathbf{y} + \mathbf{x}^T\mathbf{x}\beta) \tag{12.15}$$

Setting both derivatives to zero at the optimum, we get

$$\hat{\beta}_0 = \frac{1}{n}\mathbf{1}^T\mathbf{y} - \frac{1}{n}\mathbf{1}^T\mathbf{x}\widehat{\beta} \tag{12.16}$$

Notice that $n^{-1}\mathbf{1}^T\mathbf{y}$ is just our old friend $\overline{y}$. Similarly, $\frac{1}{n}\mathbf{1}^T\mathbf{x}$ is the $1 \times p$ matrix giving the sample means for each coordinate of x; lets call this $\overline{\mathbf{x}}$. Thus

$$\hat{\beta}_0 = \overline{y} - \overline{\mathbf{x}}\widehat{\beta} \tag{12.17}$$

and the intercept will make sure the regression surface goes through the mean of the data.

Turning to the equation for $\widehat{\beta}$,

$$0 = \hat{\beta}_0\frac{1}{n}\mathbf{x}^T\mathbf{1} - \frac{1}{n}\mathbf{x}^T\mathbf{y} + \frac{1}{n}\mathbf{x}^T\mathbf{x}\widehat{\beta} \tag{12.18}$$

At this point, let's make two moves which will simplify things later. First, notice that $\hat{\beta}_0$ is a scalar, so we can move it all the way to the right of the first term we're adding, getting

$$0 = \frac{1}{n}\mathbf{x}^T \mathbf{1}\hat{\beta}_0 - \frac{1}{n}\mathbf{x}^T \mathbf{y} + \frac{1}{n}\mathbf{x}^T \mathbf{x}\widehat{\beta} \tag{12.19}$$

Second, notice that $n^{-1}\mathbf{x}^T\mathbf{1} = \overline{\mathbf{x}}^T$. Thus

$$0 = \overline{\mathbf{x}}^T \hat{\beta}_0 - \frac{1}{n}\mathbf{x}^T \mathbf{y} + \frac{1}{n}\mathbf{x}^T \mathbf{x}\widehat{\beta} \tag{12.20}$$

Now substitute in Eq. 12.17 for $\hat{\beta}_0$:

$$\begin{aligned} 0 &= \overline{\mathbf{x}}^T(\overline{y} - \overline{\mathbf{x}}\widehat{\beta}) - \frac{1}{n}\mathbf{x}^T\mathbf{y} + \frac{1}{n}\mathbf{x}^T\mathbf{x}\widehat{\beta} && (12.21)\\ 0 &= \overline{\mathbf{x}}^T\overline{y} - \overline{\mathbf{x}}^T\overline{\mathbf{x}}\widehat{\beta} - \frac{1}{n}\mathbf{x}^T\mathbf{y} + \frac{1}{n}\mathbf{x}^T\mathbf{x}\widehat{\beta} && (12.22)\\ \frac{1}{n}\mathbf{x}^T\mathbf{x}\widehat{\beta} - \overline{\mathbf{x}}^T\overline{\mathbf{x}}\widehat{\beta} &= -\overline{\mathbf{x}}^T\overline{y} + + \frac{1}{n}\mathbf{x}^T\mathbf{y} && (12.23)\\ \left(\frac{1}{n}\mathbf{x}^T\mathbf{x} - \overline{\mathbf{x}}^T\overline{\mathbf{x}}\right)\widehat{\beta} &= \frac{1}{n}\mathbf{x}^T\mathbf{y} - \overline{\mathbf{x}}^T\overline{y} && (12.24) \end{aligned}$$

It is straight-forward to check that (Exercise 1)

$$\frac{1}{n}\mathbf{x}^T\mathbf{y} - \overline{\mathbf{x}}^T\overline{y} \tag{12.25}$$

is the $p \times 1$ matrix whose i^{th} entry is the sample covariance between X_i and Y. Similarly (Exercise 2),

$$\frac{1}{n}\mathbf{x}^T\mathbf{x} - \overline{\mathbf{x}}^T\overline{\mathbf{x}} \tag{12.26}$$

is the $p \times p$ sample variance-covariance matrix of the X_i's. (This is why I left in the seeming-redundant factors of $1/n$.)

Let us call these two matrices, respectively, $\mathbf{c}_{X,Y}$ and $\mathbf{v}_X$. Then our equation for the vector of slopes is

$$\mathbf{v}_X\widehat{\beta} = \mathbf{c}_{X,Y} \tag{12.27}$$

which of course has the solution

$$\widehat{\beta} = \mathbf{v}_X^{-1}\mathbf{c}_{X,Y} \tag{12.28}$$

In words: we find the slopes by first finding the covariance between each predictor and the response, and then multiplying by the inverse of the predictor's covariance matrices. The intercept is just a fudge-factor to make sure the regression surface goes through the mean of the data.

Taking the $n \rightarrow \infty$ limit As the sample size grows, the law of large numbers tells us $\mathbf{v}_X \rightarrow \mathrm{Var}[X]$, the true $p \times p$ variance-covariance matrix of the predictors. Similarly, $\mathbf{c}_{\mathbf{X},Y} \rightarrow \mathrm{Cov}[X,Y]$, the $p \times 1$ matrix of covariances between the predictors and the response. Hence (by continuity)

$$\widehat{\beta} \rightarrow \mathrm{Var}[X]^{-1}\mathrm{Cov}[X,Y] \tag{12.29}$$

I leave it as an exercise (3) to show that, first, under the model assumptions, the true vector of slopes β is indeed equal to $\mathrm{Var}[X]^{-1}\mathrm{Cov}[X,Y]$, and, second, that this vector of slopes would minimize the *expected* squared error (not the in-sample mean squared error).

12.3.2 Why Multiple Regression Isn't Just a Bunch of Simple Regressions

When we do multiple regression, the slopes we get for each variable aren't the same as the ones we'd get if we just did p separate simple regressions. Why not?

The book-keeping answer In §12.3.1, we saw that the slopes are determined by $\mathbf{v}_X^{-1}\mathbf{c}_{X,Y}$. If $\mathbf{v}_X^{-1}$ is diagonal, then our multiple regression *will* give the same slopes as many simple regressions. In turn, $\mathbf{v}_X^{-1}$ is diagonal if and only if $\mathbf{v}_X$ is diagonal, which means that none of the predictor variables can have any (sample) correlation with any of the others. Otherwise, minimizing the mean squared error means shifting the slopes away from what they'd be in simple regressions.

(Since $\mathbf{x}$ is called the **design matrix**, a data set where $\mathbf{v}_X$ is diagonal is said to have an **orthogonal design**. As the word suggests, this is much more common in deliberately planned experiments than in observational studies.)

The predictive answer Suppose the real model is $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$. (Nothing turns on $p = 2$, it just keeps things short.) What would happen if we did a simple regression of Y on just X_1? We know (Chapter 1) that the optimal (population) slope on X_1 should be

$$\frac{\mathrm{Cov}[X_1,Y]}{\mathrm{Var}[X_1]} \tag{12.30}$$

Let's substitute in the model equation for Y:

$$\begin{aligned}
\frac{\mathrm{Cov}[X_1,Y]}{\mathrm{Var}[X_1]} &= \frac{\mathrm{Cov}[X_1,\beta_0+\beta_1X_1+\beta_2X_2+\epsilon]}{\mathrm{Var}[X_1]} && (12.31)\\
&= \frac{\beta_1\mathrm{Var}[X_1]+\beta_2\mathrm{Cov}[X_1,X_2]+\mathrm{Cov}[X_1,\epsilon]}{\mathrm{Var}[X_1]} && (12.32)\\
&= \beta_1+\frac{\beta_2\mathrm{Cov}[X_1,X_2]+0}{\mathrm{Var}[X_1]} && (12.33)\\
&= \beta_1+\beta_2\frac{\mathrm{Cov}[X_1,X_2]}{\mathrm{Var}[X_1]} && (12.34)
\end{aligned}$$

The total covariance between X_1 and Y includes X_1's direct contribution to Y, plus the indirect contribution through correlation with X_2, and X_2's contribution to Y. (All of this applies, with subscripts swapped, to regressing Y on X_2 as well.)

Said slightly differently, when there's correlation between X_1 and X_2, we can predict (a bit of) X_2 from X_1 and vice versa. When we do simple regression, we don't care — adding up the direct and indirect relationships of Y and X_1 is fine. When we do multiple regression, we don't want to "double count" that contribution to Y, so the slopes should just reflect the relationship the response and the part of each predictor variable we couldn't have already guessed from knowing the others.

(If you're wondering, "Wait, what if there's really an X_3 but we only regressed on X_1 and X_2, wouldn't we have the same sort of problem?", congratulations — you've just discovered **omitted variable bias**.)

The geometric answer Refer again to §12.3.1. The optimal slopes are given by

$$\mathrm{Var}[X]^{-1}\mathrm{Cov}[X,Y] \tag{12.35}$$

which means that the optimal predictions are given by

$$X^T\mathrm{Var}[X]^{-1}\mathrm{Cov}[X,Y] \tag{12.36}$$

(The transpose on X is because I chose to write vectors as column matrices, and we need to make this come out a scalar.)

Now, $\mathrm{Var}[X]$ is a square, symmetric $p \times p$ matrix, so it makes sense to talk about its square root[3], i.e., a symmetric $p \times p$ matrix $\mathrm{Var}[X]^{1/2}$ such that $\mathrm{Var}[X] = \mathrm{Var}[X]^{1/2}\mathrm{Var}[X]^{1/2}$. It follows that $\mathrm{Var}[X]^{-1}$ also has a square root, $\mathrm{Var}[X]^{-1/2}$, given by $\left(\mathrm{Var}[X]^{1/2}\right)^{-1}$. Thus we can say that the optimal predictions are given by

$$\begin{aligned}
X^T\mathrm{Var}[X]^{-1}\mathrm{Cov}[X,Y] &= X^T\mathrm{Var}[X]^{-1/2}\mathrm{Var}[X]^{-1/2}\mathrm{Cov}[X,Y] && (12.37)\\
&= (\mathrm{Var}[X]^{-1/2}X)^T\mathrm{Cov}\left[\mathrm{Var}[X]^{-1/2}X, Y\right] && (12.38)
\end{aligned}$$

By the rules for algebra with variances,

$$\begin{aligned}
\mathrm{Var}\left[\mathrm{Var}[X]^{-1/2}X\right] &= \mathrm{Var}[X]^{-1/2}\mathrm{Var}[X]\mathrm{Var}[X]^{-1/2} && (12.39)\\
&= \mathrm{Var}[X]^{-1/2}\mathrm{Var}[X]^{1/2}\mathrm{Var}[X]^{1/2}\mathrm{Var}[X]^{-1/2} = \mathbf{I} && (12.40)
\end{aligned}$$

Multiplying a vector by a matrix rotates and stretches the coordinate system for the vector. Multiplying X by $\mathrm{Var}[X]^{-1/2}$ rotates and stretches the coordinates so that all the components of X are uncorrelated with each other, and they all have variance 1. The point of the $\mathrm{Var}[X]^{-1}$ in the formula for the regression slopes is that it, implicitly, finds the new coordinate system where the predictors are uncorrelated, and then does a bunch of simple regressions.

[3]For instance, we know from the "spectral" or "eigendecomposition" theorem in linear algebra that such a matrix can be written as $U\Lambda U^T$, where U is the $p \times p$ matrix whose columns are the eigenvectors, and Λ is the diagonal matrix of eigenvalues.

12.3.3 Point Predictions and Fitted Values

Just as with simple regression, the vector of fitted values $\widehat{\mathbf{m}}$ is linear in $\mathbf{y}$, and given by the hat matrix:

$$\begin{aligned} \widehat{\mathbf{m}} &= \mathbf{x}\widehat{\beta} && (12.41)\\ &= \mathbf{x}(\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{y} && (12.42)\\ &= \mathbf{H}\mathbf{y} && (12.43)\end{aligned}$$

All of the interpretations given of the hat matrix in the previous chapter still apply. The hat matrix remains square ($n \times n$), symmetric ($\mathbf{H}^T = \mathbf{H}$) and idempotent ($\mathbf{HH} = \mathbf{H}$). One important property needs to be added for this general case: the trace is the number of coefficients, $\operatorname{tr}\mathbf{H} = p+1$ (Exercise 6).

12.4 Properties of the Estimates

We will only look at the most basic properties of bias and variance here, deferring the full sampling distribution, and confidence sets, to next time.

The fundamental observation is the following. Let's hold $\mathbf{x}$ fixed, and let $\mathbf{Y}$ vary randomly. Since

$$\widehat{\beta} = (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{Y} \qquad (12.44)$$

and

$$\mathbf{Y} = \mathbf{x}\beta + \epsilon \qquad (12.45)$$

we have

$$\widehat{\beta} = (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{x}\beta + (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\epsilon = \beta + (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\epsilon \qquad (12.46)$$

12.4.1 Bias

This is straight-forward:

$$\begin{aligned} \mathbb{E}\left[\widehat{\beta}|\mathbf{x}\right] &= \mathbb{E}\left[\beta + (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\epsilon|\mathbf{x}\right] && (12.47)\\ &= \beta + (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbb{E}\left[\epsilon|\mathbf{x}\right] && (12.48)\\ &= \beta && (12.49)\end{aligned}$$

Thus, the least squares estimate of the general linear model's coefficients is conditionally unbiased, no matter what p is.

Notice that we needed to use one of the modeling assumptions to get this: if the true regression function wasn't linear, we couldn't say $\mathbb{E}[\epsilon|\mathbf{x}] = 0$.

12.4.2 Variance and Standard Errors

This needs a little more work.

$$\begin{aligned}
\mathrm{Var}\left[\widehat{\beta}|\mathbf{x}\right] &= \mathrm{Var}\left[\beta + (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\epsilon|\mathbf{x}\right] && (12.50)\\
&= \mathrm{Var}\left[(\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\epsilon|\mathbf{x}\right] && (12.51)\\
&= (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathrm{Var}\left[\epsilon|\mathbf{x}\right]\mathbf{x}(\mathbf{x}^T\mathbf{x})^{-1} && (12.52)\\
&= (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\sigma^2\mathbf{I}\mathbf{x}(\mathbf{x}^T\mathbf{x})^{-1} && (12.53)\\
&= \sigma^2(\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{x}(\mathbf{x}^T\mathbf{x})^{-1} && (12.54)\\
&= \sigma^2(\mathbf{x}^T\mathbf{x})^{-1} && (12.55)
\end{aligned}$$

Again, this is true whatever p might be.

To understand this a little better, let's re-write it slightly:

$$\mathrm{Var}\left[\widehat{\beta}|\mathbf{x}\right] = \frac{\sigma^2}{n}(n^{-1}\mathbf{x}^T\mathbf{x})^{-1} \tag{12.56}$$

The first term, σ^2/n, is what we're familiar with from the simple linear model. As n grows, we expect the entries in $\mathbf{x}^T\mathbf{x}$ to be increasing in magnitude, since they're sums over all n data points; dividing all entries in the matrix by n compensates for this. If the sample covariances between all the predictor variables were 0, when we took the inverse we'd get $1/s^2_{X_i}$ down the diagonal (except for the top of the diagonal), just as we got $1/s^2_X$ in the simple linear model.

12.5 Collinearity

I have been silently assuming that $(\mathbf{x}^T\mathbf{x})^{-1}$ exists, that $\mathbf{x}^T\mathbf{x}$ is "invertible" or "non-singular". There are a number of equivalent conditions for a matrix to be invertible:

1. Its determinant is non-zero.
2. It is of "full column rank", meaning all of its columns are linearly independent[4].
3. It is of "full row rank", meaning all of its rows are linearly independent.

The equivalence of these conditions are mathematical facts, proved in linear algebra; I will not re-prove them here.

What does this amount to in terms of our data? It means (Exercise 5) that the variables must be linearly independent *in our sample*. That is, there must not be any set of constants $a_0, a_1, \ldots a_p$ where, for *all* rows i,

$$a_0 + \sum_{j=1}^{p} a_j x_{ij} = 0 \tag{12.57}$$

[4]Recall that a set of vectors is linearly independent if no linear combination of them is exactly zero.

This, in other words, means that **x** must be of full column rank.

To understand why linearly dependence among variables is a problem, take an easy case, where two predictors, say X_1 and X_2, are exactly equal to each other. It's then not surprising that we don't have any way of estimating their coefficients. If we get one set of predictions with coefficients β_1, β_2, we'd get exactly the same predictions from $\beta_1 + \gamma$, $\beta_2 - \gamma$, no matter what γ might be. If there are other exact linear relations among two variables, we can similarly trade off their coefficients against each other, without any change in anything we can observe. If there are exact linear relationships among more than two variables, all of their coefficients become ill-defined.

We will come back in a few chapters to what to do when faced with collinearity. For now, we'll just mention a few clear situations:

- If $n < p + 1$, the data are collinear.
- If one of the predictor variables is constant, the data are collinear.
- If two of the predictor variables are proportional to each other, the data are collinear.
- If two of the predictor variables are otherwise linearly related, the data are collinear.

While it's important to double-check for these, for right now, we'll hope it doesn't happen. That does mean, however, that we need to look and see whether it *is* happening.

12.6 R Practicalities

12.6.1 `lm`

`lm` works in almost the same way as for simple linear models. Let's look at the model from the last data analysis project:

```
mobility <- read.csv("http://www.stat.cmu.edu/~cshalizi/mreg/15/dap/1/mobility.csv")
```

The only real change is that we need to tell `lm`, through the formula, what all the predictor variables are; we do this with + signs:

```
# Fit a model with three predictors
mob.lm <- lm(Mobility ~ Commute + Latitude + Longitude, data = mobility)
```

The order of the predictor variables only matters for the order in which the coefficients will be listed. All of the utility functions we already know still work, in exactly the same way:

```
# Basic print-out:
print(mob.lm)
##
```

```
## Call:
## lm(formula = Mobility ~ Commute + Latitude + Longitude, data = mobility)
##
## Coefficients:
## (Intercept)      Commute     Latitude    Longitude
##  -3.136e-02    2.010e-01    9.383e-04   -4.305e-05
# Coefficients:
coefficients(mob.lm)
##   (Intercept)       Commute      Latitude     Longitude
## -3.136000e-02  2.009679e-01  9.383055e-04 -4.304546e-05
# Confidence intervals for parameters:
confint(mob.lm)
##                     2.5 %        97.5 %
## (Intercept) -0.0563094963 -0.0064104992
## Commute      0.1738437953  0.2280920687
## Latitude     0.0003580771  0.0015185339
## Longitude   -0.0002827799  0.0001966889
# Fitted values:
head(fitted(mob.lm))
##          1          2          3          4          5          6
## 0.07172344 0.06156703 0.07867982 0.06006085 0.06464329 0.06943562
# Residuals:
head(residuals(mob.lm))
##            1            2            3            4            5            6
## -0.009524631 -0.007915094 -0.006044680 -0.003779634 -0.019842494 -0.017599771
```

`summary` is also basically the same, but slightly more elaborate.

```
summary(mob.lm)
##
## Call:
## lm(formula = Mobility ~ Commute + Latitude + Longitude, data = mobility)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.17583 -0.02222 -0.00586  0.01758  0.32290
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.136e-02  1.271e-02  -2.468  0.01383 *
## Commute      2.010e-01  1.382e-02  14.546  < 2e-16 ***
## Latitude     9.383e-04  2.956e-04   3.175  0.00156 **
## Longitude   -4.305e-05  1.221e-04  -0.353  0.72456
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04227 on 725 degrees of freedom
## Multiple R-squared:  0.3583,Adjusted R-squared:  0.3557
## F-statistic: 134.9 on 3 and 725 DF,  p-value: < 2.2e-16
```

This lists t-tests for every coefficient; we will go exactly how to interpret those next time.

As usual, it is *much better* to use a formula with just column names and a `data` argument than to hard-code in particular vectors.

12.6.2 `predict`

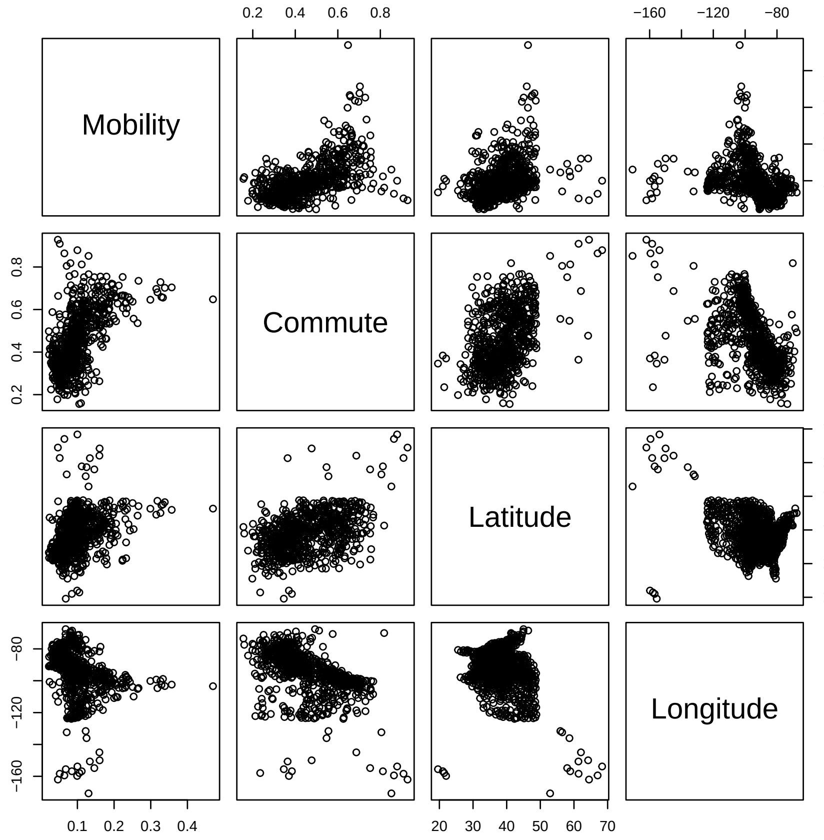
`predict` also works in exactly the same way, only we need to give a data frame with columns for each of the predictor variables:

```
predict(mob.lm, newdata = data.frame(Commute = 0.5, Latitude = 40.35, Longitude = -79.92))  # Where
##         1
## 0.1104248
```

Confidence intervals for conditional means, and prediction intervals, work in just the same way as before.

12.6.3 Exploratory Plots

While we will go over the diagnostic plots next time, some exploratory plots are needed at this point. The simplest thing to do is a bivariate scatter-plot for every pair of variables. You *could* do this by writing `plot` umpteen times, but this is such a common task that there's a useful R function to make all possible scatterplots, called `pairs` (Figure 12.1).



```
pairs(~Mobility + Commute + Latitude + Longitude, data = mobility)
```

FIGURE 12.1: *Example of using* `pairs`*: the formula has an empty left-hand side (because there isn't really a distinguished response variable), and all the variables we want to plot on the right-hand side. If we left out the formula, we'd get plots of all variables against all others: why isn't that sensible here? What would happen if we used the formula* `Mobility ∼ Commute + Latitude + Longitude`*?*

12.7 Exercises

1. Show that
$$\frac{1}{n}\mathbf{x}^T\mathbf{y} - \overline{\mathbf{x}}^T\overline{y} \tag{12.58}$$
is the $p \times 1$ matrix whose i^{th} entry is the sample covariance between X_i and Y.

2. Show that
$$\frac{1}{n}\mathbf{x}^T\mathbf{x} - \overline{\mathbf{x}}^T\overline{\mathbf{x}} \tag{12.59}$$
is the $p \times p$ matrix whose i, j entry is the sample covariance between X_i and X_j.

3.) Show the following:
   (a) That in the multiple-regression model, the true vector of slopes β equals $\text{Var}[X]^{-1}\text{Cov}[X,Y]$.
   (b) That this vector of slopes minimizes the *expected* squared error.

4. Assume $p = 2$. Work out $n^{-1}\mathbf{x}^T\mathbf{x}$ and $(n^{-1}\mathbf{x}^T\mathbf{x})^{-1}$ in terms of $\overline{x_1}$, $\overline{x_2}$, $\overline{x_1 x_2}$, $\overline{x_1^2}$ and $\overline{x_2^2}$.

5. (a) Show if $\mathbf{x}$ is of full column rank, than $\mathbf{x}^T\mathbf{x}$ is also of full rank.
   (b) Show that if $\mathbf{x}^T\mathbf{x}$ is not of full rank, then $\mathbf{x}$ must be of less than full column rank.

6. Suppose that we have p predictor variables in a multiple linear regression. Show that $\text{tr}\,\mathbf{H}$, the trace of the hat matrix, is exactly $p + 1$.

   Hint: Use the "cyclic rule" for traces: for any three matrices $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$, where $\mathbf{abc}$ is a square matrix, $\text{tr}(\mathbf{abc}) = \text{tr}(\mathbf{bca}) = \text{tr}(\mathbf{cab})$.

   Alternative hint (harder): First, show that $\mathbf{H}$ is idempotent, $\mathbf{HH} = \mathbf{H}$. Next, show that the only possible eigenvalues of any idempotent matrix are 0 and 1. Finally, show that $\mathbf{H}$ has exactly $p + 1$ non-zero eigenvalues.

Chapter 13

Diagnostics and Inference for Multiple Linear Regression

13.1 Lighting Review of Multiple Linear Regression

In the multiple linear regression model, we assume that the response Y is a linear function of all the predictors, plus a constant, plus noise:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots \beta_p X_p + \epsilon_i \tag{13.1}$$

We assume nothing about the (marginal or joint) distributions of the X_i, but we do assume that $\mathbb{E}[\epsilon|X] = 0$, that $\mathrm{Var}[\epsilon|X] = \sigma^2$, and that ϵ_i is uncorrelated across data points i. In matrix form, the model is

$$\mathbf{Y} = \mathbf{X}\beta + \epsilon \tag{13.2}$$

where $\mathbf{X}$ includes an initial column of all 1s.

When we add the Gaussian noise assumption, we are making all of the assumptions above, and further assuming that

$$\epsilon \sim MVN(0, \sigma^2 \mathbf{I}) \tag{13.3}$$

independently of $\mathbf{X}$.

The least squares estimate of the coefficients is

$$\widehat{\beta} = (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{y} \tag{13.4}$$

Under the Gaussian noise assumption, this is also the maximum likelihood estimate.

The fitted values (i.e., estimates of the conditional means at data points used to estimate the model) come from the "hat" or "influence" matrix:

$$\widehat{\mathbf{m}} = \mathbf{x}\widehat{\beta} = \mathbf{H}\mathbf{y} \tag{13.5}$$

which is symmetric and idempotent. The vector of residuals is

$$\mathbf{e} = (\mathbf{I} - \mathbf{H})\mathbf{y} \tag{13.6}$$

and $\mathbf{I} - \mathbf{H}$ is also symmetric and idempotent.

The expected mean squared error, which is the maximum likelihood estimate of σ^2, has a small negative bias:

$$\mathbb{E}\left[\hat{\sigma}^2\right] = \mathbb{E}\left[\frac{1}{n}\mathbf{e}^T\mathbf{e}\right] = \sigma^2\frac{n-p-1}{n} = \sigma^2\left(1 - \frac{p+1}{n}\right) \tag{13.7}$$

Since $\mathbf{H}\mathbf{x}\beta = \mathbf{x}\beta$, the residuals are also

$$\mathbf{e} = (\mathbf{I} - \mathbf{H})\epsilon \tag{13.8}$$

hence

$$\mathbb{E}[\mathbf{e}] = \mathbf{0} \tag{13.9}$$

and

$$\mathrm{Var}[\mathbf{e}] = \sigma^2(\mathbf{I} - \mathbf{H}) \tag{13.10}$$

Under the Gaussian noise assumption, $\widehat{\beta}$, $\widehat{\mathbf{m}}$ and $\mathbf{e}$ all have Gaussian distributions (about which more below, §13.3.1).

13.1.1 Point Predictions

Say that $\mathbf{x}'$ is the $m \times (p+1)$ dimensional matrix storing the values of the predictor variables at m points where we want to make predictions. (These may or may not include points we used to estimate the model, and m may be bigger, smaller or equal to n.) Similarly, let $\mathbf{Y}'$ be the $m \times 1$ matrix of random values of Y at those points. The point predictions we want to make are

$$\mathbb{E}\left[\mathbf{Y}'|\mathbf{X}' = \mathbf{x}'\right] = \mathbf{m}(\mathbf{x}') = \mathbf{x}'\beta \tag{13.11}$$

and we *estimate* this by

$$\widehat{\mathbf{m}}(\mathbf{x}') = \mathbf{x}'\widehat{\beta} \tag{13.12}$$

which is to say

$$\widehat{\mathbf{m}}(\mathbf{x}') = \mathbf{x}'(\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{y} \tag{13.13}$$

(It's easy to verify that when $\mathbf{x}' = \mathbf{x}$, this reduces to $\mathbf{H}\mathbf{y}$.)

Notice that the point predictions we make *anywhere* are always weighted sums (linear combinations) of the values of the response we happened to observe when we estimated the model. The weights just depend on the values of the predictors at the original data points, and at the points where we'll be making predictions — changing the responses doesn't change those weights.

13.2 Diagnostics for Multiple Linear Regression

Before proceeding to detailed statistical inference, we need to check our modeling assumptions, which means we need diagnostics.

13.2.1 Plot All the Things!

All of the plots we learned how to do for simple linear regression remain valuable:

1. *Plot the residuals against the predictors.* This now means p distinct plots, of course. Each of them should show a flat scatter of points around `0` (because $\mathbb{E}[\epsilon|X_i] = 0$), of roughly constant width (because $\mathrm{Var}[\epsilon|X_i] = \sigma^2$). Curvature or steps to this plot is a sign of potential nonlinearity, or of an omitted variable. Changing width is a potential sign of non-constant variance.

2. *Plot the squared residuals against the predictors.* Each of these p plots should show a flat scatter of points around $\hat{\sigma}^2$.

3. *Plot the residuals against the fitted values.* This is an extra plot, redundant when we only have one predictor (because the fitted values were linear in the predictor).

4. *Plot the squared residuals against the fitted values.*

5. *Plot the residuals against coordinates.* If observations are dated, time-stamped, or spatially located, plot the residuals as functions of time, or make a map. If there is a meaningful order to the observations, plot residuals from successive observations against each other. Because the ϵ_i are uncorrelated, all of these plots should show a lack of structure.

6. *Plot the residuals' distribution against a Gaussian.*

Out-of-sample predictions, with either random or deliberately selected testing sets, also remain valuable.

13.2.1.1 Collinearity

A linear dependence between two (or more) columns of the $\mathbf{x}$ matrix is called **collinearity**, and it keeps us from finding a solution by least squares. (In fact, collinearity at the population level makes the coefficients ill-defined, not just impossible to estimate.) Collinearity between a pair of variables will show up in a pairs plot as an exact straight line. Collinearity among more than two variables will not. For instance, if $X_3 = (X_1 + X_2)/2$, we can't include all three variables in a regression, but we'd not see that from any of the pairs.

Computationally, collinearity will show up in the form of the determinant of $\mathbf{x}^T\mathbf{x}$ being zero. Equivalently, the smallest eigenvalue of $\mathbf{x}^T\mathbf{x}$ will be zero. If `lm` is given a collinear set of predictor variables, it will sometimes give an error messages, but more often it will decide not to estimate one of the collinear variables, and return an `NA` for the offending coefficient.

We will return to the subject of collinearity in Chapter 15

```
# Minimal simulation of interactions Model: Y = sin(X1*X2) + noise
X <- matrix(runif(200), ncol = 2)
Y <- sin(X[, 1] * X[, 2]) + rnorm(200, 0, 0.1)
df <- data.frame(Y = Y, X1 = X[, 1], X2 = X[, 2])
missed.interact <- lm(Y ~ X1 + X2, data = df)
```

FIGURE 13.1: *Simulating data from the model $Y = \sin X_1 X_2 + \epsilon$, to illustrate detecting interactions. Self-checks: what is the distribution of X_1 and X_2? what is σ^2?*

13.2.1.2 Interactions

Another possible complication for multiple regression which we didn't have with the simple regression model is that of *interactions* between variables. One of our assumptions is that each variable makes a distinct, additive contribution to the response, and the size of this contribution is completely insensitive to the contributions of other variables. If this is *not* true — if the relationship between Y and X_i changes depending on the value of another predictor, X_j — then there is an **interaction** between them.

There are several ways of looking for interactions. We will return to this subject in Chapter 17, but, for now, I'll stick with describing some diagnostic procedures.

Sub-divide and re-estimate The simplest thing to do, if you suspect an interaction between X_i and X_j, is to sub-divide the data based on the value of X_j, into two or more parts, and re-estimate the model. If there is no interaction, the coefficient on X_i should be the same, up to estimation error, in each part of the data. (That is, there should be no significant difference in the estimated coefficients.) While in principle straightforward, drawbacks to this include having to guess how to sub-divide the data (into two parts? three? more?), and at what values of X_j to make the cuts.

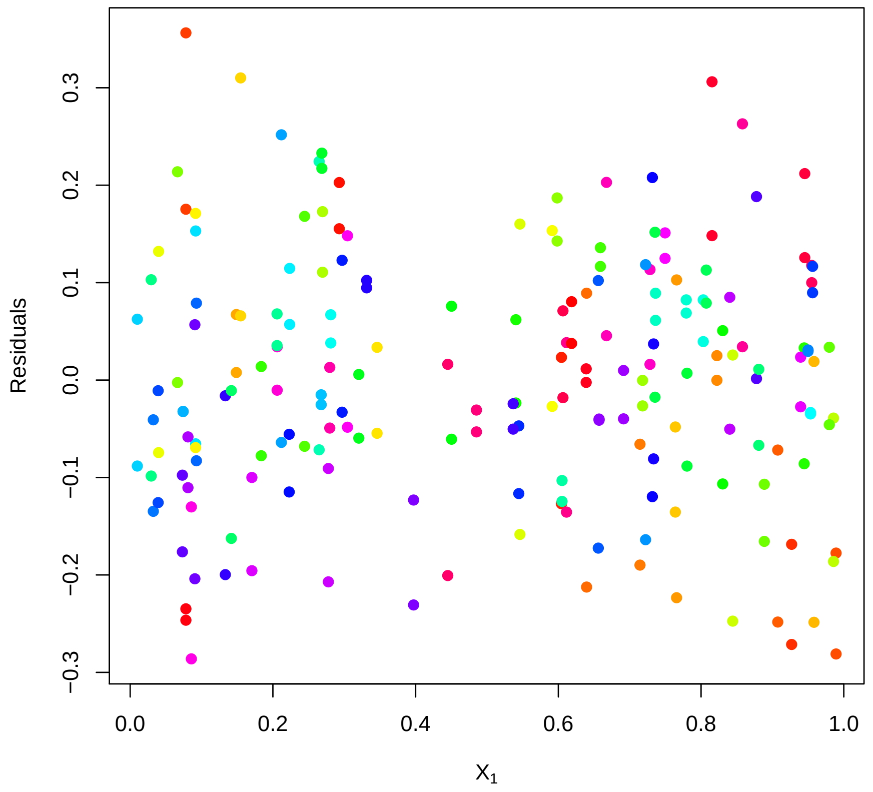
Scatterplot with color or symbols A more visual alternative is to plot the residuals against X_i, as usual, but to give each point a color which varies continuously with the value of X_j. In the absence of interactions, there should be no pattern to the colors. If there are interactions, however, we could predict what the residuals will be from knowing both variables, so we should tend to see similarly-colored regions in the plot.

If color is not available, a similar effect can be obtained by using different plotting symbols, corresponding to different values of X_j.

3D Plots

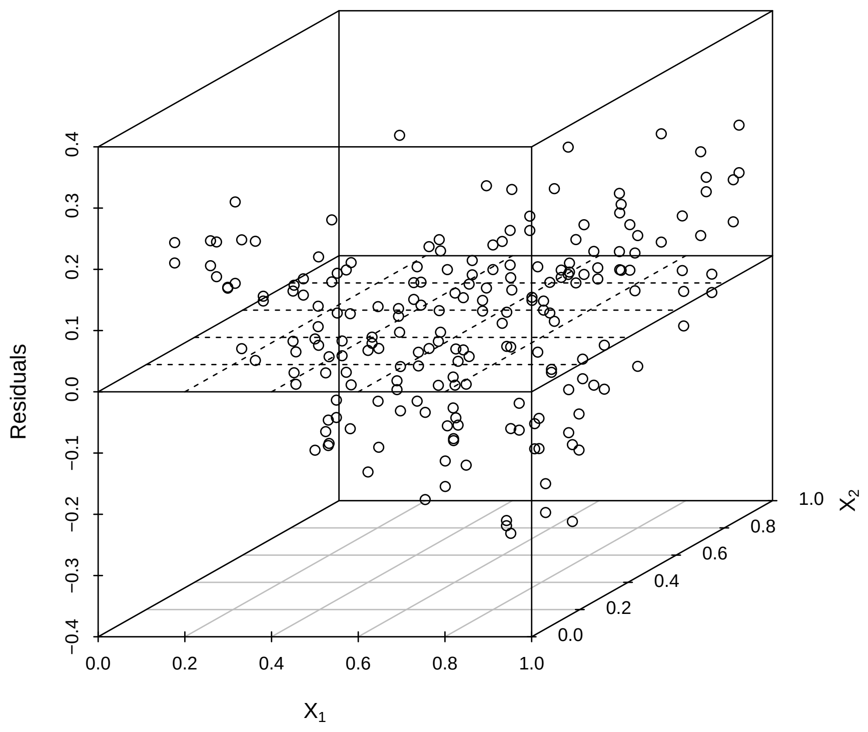
```
coefficients(summary(lm(Y ~ X1 + X2, data = df, subset = which(df$X2 < median(df$X2)))))
##                Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) -0.08750095 0.03203652 -2.731288 7.495314e-03
## X1           0.16382609 0.04505160  3.636410 4.449144e-04
## X2           0.49163582 0.08118309  6.055890 2.645673e-08
coefficients(summary(lm(Y ~ X1 + X2, data = df, subset = which(df$X2 > median(df$X2)))))
##               Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) -0.2388972 0.05086796 -4.696418 8.718322e-06
## X1           0.6225676 0.03652214 17.046308 6.188960e-31
## X2           0.3837526 0.06733542  5.699120 1.294438e-07
```

FIGURE 13.2: *Here we have sub-setted the data based on the value of the second predictor (dividing it, somewhat arbitrarily, at its median). Notice that the difference in the two coefficients for X_1 is much larger than their standard errors. Can you give a significance level for the difference in means?*



```
# Create a vector of gradually-changing colors, with one entry for each data
# point
the.colors <- rainbow(n = nrow(df))
# For each data point, see how it ranks according to X2, from smallest (1)
# to largest
the.ranks <- rank(df$X2)
# Plot residuals vs. X1, colored according to X2 Defining the color and rank
# vectors makes this next line a bit less mysterious, but it's not
# necessary; this could all be a one-liner.
plot(df$X1, residuals(missed.interact), pch = 19, col = the.colors[the.ranks],
    xlab = expression(X[1]), ylab = "Residuals")
```

FIGURE 13.3: *Plotting residuals from the linear model against X_1, with the color of the point set by the value of X_2. Notice the clumping of points with similar colors: this means that knowing both X_1 and X_2 lets us predict the residual. Horizontal bands of the same color, on the other hand, would show that X_2 helped predict the residuals but X_1 did not, pointing to a mis-specification for the dependence of Y on X_2.*



```
library(scatterplot3d)
# Make a 3D scatterplot of residuals against the two predictor variables
s3d <- scatterplot3d(x = df$X1, y = df$X2, z = residuals(missed.interact), tick.marks = TRUE,
    label.tick.marks = TRUE, xlab = expression(X[1]), ylab = expression(X[2]),
    zlab = "Residuals")
# Add a plane with intercept 0 and both slopes also 0, for visual reference
s3d$plane3d(c(0, 0, 0), lty.box = "solid")
```

FIGURE 13.4: *Residuals (vertical axis) vs. predictor variables. Notice that there are regions where the residuals are persistent positive or negative, but that these are defined by the value of* both *variables, not one or the other alone.*

13.2.2 Remedies

All of the remedies for model problems we discussed earlier, for the simple linear model, are still available to us.

Transform the response We can change the response variable from Y to $g(Y)$, in the hope that the assumptions of the linear-Gaussian model are more nearly satisfied for this new variable. That is, we hope that

$$g(Y) = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p + \epsilon, \ \epsilon \sim N(0, \sigma 0^2 \tag{13.14}$$

The Box-Cox method, if you want to use it, will work as well as it did for simple linear models. Computationally, we'd just fill the $n \times 1$ response matrix with $[g(y_1)\ g(y_2)\ \ldots g(y_n)]^T$, and proceed as with any other multiple regression.

However, see the handout on transformations for cautions on interpretation after such transformations.

Transform the predictors We can also transform each of the predictors, making the model

$$Y = \beta_0 + \beta_1 f_1(X_1) + \ldots \beta_p f_p(X_p) + \epsilon, \ \epsilon \sim N(0, \sigma^2) \tag{13.15}$$

As the notation suggests, each X_i could be subject to a different transformation. Again, it's just a matter of what we put in the columns of the **x** matrix before solving for $\widehat{\beta}$. Again, see the handout on transformations for cautions on interpretations.

(A model of this form is called an **additive** model; in 402 we will look extensively at how they can be estimated, by automatically searching for near-optimal transformations.)

An alternative is to transform, not each predictor variable, but their linear combination:

$$Y = h\left(\beta_0 + \beta_1 X_1 + \ldots \beta_p X_p\right) + \epsilon, \ \epsilon \sim N(0, \sigma^2) \tag{13.16}$$

This is called a "single index" model, because there is only one combination of the predictors, the weighted sum $\beta_1 X_1 + \ldots \beta_p X_p$, which matters to the response. Notice that this is *not* the same model as the transform-Y model, even if $h = g^{-1}$, because of the different role of the noise.

Changing the variables used One option which is available to us with multiple regression is to add in new variables, or to remove ones we're already using. This should be done carefully, with an eye towards satisfying the model assumptions, rather than blindly increasing some score. We will discuss this extensively in Chapters 19 and 22.

13.2.3 Plot *All* the Things?

There is one important caution about exuberant diagnostics plotting. This is that the more checks you run, the bigger the chance that you will find something which looks weird *just by chance*. If we were doing formal hypothesis tests, and insisted on a uniform false positive rate of α, then after running r tests, we'd expect to make $\approx r\alpha$ rejections, *even if all of our null hypotheses are true*. (Why?) If you are doing lots of diagnostic plots — say, 20 or 30 or more — it becomes a very good idea to do some randomization to see whether the *magnitude* of the bad-looking things you're seeing is about what you should be anticipating from one plot or another, even if everything was absolutely fine.

13.3 Inference for Multiple Linear Regression

Unless I say otherwise, all results in this section presume that all of the modeling assumptions, Gaussian noise very much included, are correct. Also, all distributions stated are conditional on $\mathbf{x}$.

13.3.1 Sampling Distributions

As in the simple linear model, the sampling distributions are the basis of all inference.

13.3.1.1 Gaussian Sampling Distributions

Gaussian distribution of coefficient estimators In the simple linear model, because the noise ϵ is Gaussian, and the coefficient estimators were linear in the noise, $\widehat{\beta}_0$ and $\widehat{\beta}_1$ were also Gaussian. This remains true in for Gaussian multiple linear regression models:

$$\begin{aligned}
\widehat{\beta} &= (\mathbf{x}^T\mathbf{x})\mathbf{x}^T\mathbf{Y} && (13.17)\\
&= (\mathbf{x}^T\mathbf{x})\mathbf{x}^T(\mathbf{x}\beta+\epsilon) && (13.18)\\
&= \beta+(\mathbf{x}^T\mathbf{x})\mathbf{x}^T\epsilon && (13.19)
\end{aligned}$$

Since $(\mathbf{x}^T\mathbf{x})\mathbf{x}^T\epsilon$ is a constant times a Gaussian, it is also a Gaussian; adding on another Gaussian still leaves us with a Gaussian. We saw the expectation and variance last time, so

$$\widehat{\beta} \sim MVN(\beta, \sigma^2(\mathbf{x}^T\mathbf{x})^{-1}) \quad (13.20)$$

It follows that

$$\widehat{\beta}_i \sim N\left(\beta_i, \sigma^2(\mathbf{x}^T\mathbf{x})^{-1}_{ii}\right) \quad (13.21)$$

Gaussian distribution of estimated conditional means The same logic applies to the estimates of conditional means. In §13.1.1, we saw that the estimated conditional means at new observations $\mathbf{x}'$ are given by

$$\widehat{\mathbf{m}}(\mathbf{x}') = \mathbf{x}'(\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{y} \quad (13.22)$$

so (Exercise)

$$\widehat{\mathbf{m}}(\mathbf{x}') \sim MVN(\mathbf{x}'\beta, \sigma^2\mathbf{x}'(\mathbf{x}^T\mathbf{x})^{-1}(\mathbf{x}')^T) \quad (13.23)$$

Gaussian distribution of fitted values Eq. 13.23 simplifies for the special case of the fitted values, i.e., the estimated conditional means on the original data.

$$\widehat{\mathbf{m}}(\mathbf{x}') \sim MVN(\mathbf{x}\beta, \sigma^2\mathbf{H}) \tag{13.24}$$

Gaussian distribution of residuals Similarly, the residuals have a Gaussian distribution:

$$\mathbf{e} \sim MVN(0, \sigma^2(\mathbf{I}-\mathbf{H})) \tag{13.25}$$

13.3.1.2 $\hat{\sigma}^2$ and Degrees of Freedom

The in-sample mean squared error $\hat{\sigma}^2 = n^{-1}\mathbf{e}^T\mathbf{e}$ has the distribution

$$\frac{n\hat{\sigma}^2}{\sigma^2} \sim \chi^2_{n-(p+1)} \tag{13.26}$$

I won't prove this here, because it involves the same sort of tedious manipulations of Gaussians as I evaded in showing the special-case χ^2_{n-2} result for simple linear models. To give a hint of what's going on, though, I'll make two (related) observations.

Constraints on the residuals The residuals are not all independent of each other. In the case of the simple linear model, the fact that we estimated the model by least squares left us with two constraints, $\sum_i e_i = 0$ and $\sum_i e_i x_i = 0$. If we had only one constraint, that would let us fill in the last residual if we knew the other $n-1$ residuals. Having two constraints meant that knowing any $n-2$ residuals determined the remaining two.

We got those constraints from the normal or estimating equations, which in turn came from setting the derivative of the mean squared error (or of the log-likelihood) to zero. In the multiple regression model, when we set the derivative to zero, we get the matrix equation

$$\mathbf{x}^T(\mathbf{y}-\mathbf{x}\widehat{\beta}) = 0 \tag{13.27}$$

But the term in parentheses is just $\mathbf{e}$, so the equation is

$$\mathbf{x}^T\mathbf{e} = 0 \tag{13.28}$$

Expanding out the matrix multiplication,

$$\left[\begin{array}{c} \sum_i e_i \\ \sum_i x_{i1}e_i \\ \vdots \\ \sum_i x_{ip}e_i \end{array}\right] = \left[\begin{array}{c} 0 \\ 0 \\ \vdots \\ 0 \end{array}\right] \tag{13.29}$$

Thus the residuals are subject to $p+1$ linear constraints, and knowing any $n-(p+1)$ of them will fix the rest.

Geometric interpretation of constraints The vector of residuals $\mathbf{e}$ is a point in an n-dimensional space. As a random vector, without any constraints it could lie anywhere in that space, as, for instance, ϵ can. The constraints, however, for it to live in a lower-dimensional subspace, specifically, a space of dimension $n-(p+1)$.

Bias of $\hat{\sigma}^2$ As more of a formal manipulation, when we look at the expectation of $\hat{\sigma}^2$, we get

$$\begin{aligned}
\mathbb{E}\left[\hat{\sigma}^2\right] &= \frac{1}{n}\mathbb{E}\left[\mathbf{e}^T\mathbf{e}\right] && (13.30)\\
&= \frac{1}{n}\mathbb{E}\left[((\mathbf{I}-\mathbf{H})\mathbf{e})^T((\mathbf{I}-\mathbf{H})\mathbf{e})\right] && (13.31)\\
&= \frac{1}{n}\mathbb{E}\left[\mathbf{e}^T(\mathbf{I}^T-\mathbf{H}^T)(\mathbf{I}-\mathbf{H})\mathbf{e}\right] && (13.32)\\
&= \frac{1}{n}\mathbb{E}\left[\mathbf{e}^T(\mathbf{I}-\mathbf{H}-\mathbf{H}^T+\mathbf{H}^T\mathbf{H})\mathbf{e}\right] && (13.33)\\
&= \frac{1}{n}\mathbb{E}\left[\mathbf{e}^T(\mathbf{I}-\mathbf{H})\mathbf{e}\right] && (13.34)
\end{aligned}$$

using the easily-checked facts that $\mathbf{H}=\mathbf{H}^T$, and that $\mathbf{H}^2=\mathbf{H}$. We've therefore reduced the expectation to a quadratic form, and so (Chapter 11)

$$\begin{aligned}
\mathbb{E}\left[\hat{\sigma}^2\right] &= \frac{1}{n}\operatorname{tr}((\mathbf{I}-\mathbf{H})\operatorname{Var}[\mathbf{e}]) && (13.35)\\
&= \frac{1}{n}\operatorname{tr}((\mathbf{I}-\mathbf{H})\sigma^2(\mathbf{I}-\mathbf{H})) && (13.36)\\
&= \frac{\sigma^2}{n}\operatorname{tr}(\mathbf{I}-\mathbf{H})^2 && (13.37)\\
&= \frac{\sigma^2}{n}\operatorname{tr}(\mathbf{I}-\mathbf{H}) && (13.38)
\end{aligned}$$

since we've just seen that $(\mathbf{I}-\mathbf{H})^2=(\mathbf{I}-\mathbf{H})$, and (Eq. 13.10) $\operatorname{Var}[\mathbf{e}]=\sigma^2(\mathbf{I}-\mathbf{H})$. Making one last push,

$$\mathbb{E}\left[\hat{\sigma}^2\right]=\frac{\sigma^2}{n}(n-p-1) \tag{13.39}$$

since $\operatorname{tr}\mathbf{I}=n$ while (as you proved in the homework) $\operatorname{tr}H=p+1$.

13.3.2 t Distributions for Coefficient and Conditional Mean Estimators

From Eq. 13.21, it follows that

$$\frac{\hat{\beta}_i-\beta_i}{\sigma^2(\mathbf{x}^T\mathbf{x})^{-1}_{ii}} \sim N(0,1) \tag{13.40}$$

This would be enough to let us do hypothesis tests and form confidence intervals, if only we knew σ^2, Since that's estimated itself, and $\hat{\sigma}^2$ has a distribution derived from a χ^2_{n-p-1}, we can go through the same arguments we did in the simple linear model case to get t distributions. Specifically,

$$\frac{\hat{\beta}_i - \beta_i}{\widehat{\mathrm{se}}\left[\hat{\beta}_i\right]} \sim t_{n-p-1} \tag{13.41}$$

The same applies to the estimated conditional means, and to the distribution of a new Y' around the estimated conditional mean (in a prediction interval). Thus, all the theory we did for parametric and predictive inference in the simple model carries over, just with a different number of degrees of freedom.

As with the simple model, $t_{n-p-1} \rightarrow N(0,1)$, so t statistics approach z statistics as the sample size grows.

13.3.3 What, Exactly, Is R Testing?

The `summary` function lists a p-value for each coefficient in a linear model. For each coefficient, say β_i, this is the p-value in testing the hypothesis that $\beta_i = 0$. It is important to be very clear about exactly what this means.

The hypothesis being tested is "Y is a linear function of all of the X_i, $i \in 1:p$, with constant-variance, independent Gaussian noise, and it just so happens that $\beta_i = 0$". Since, as we saw in Chapter 12, the optimal coefficients for each predictor variable depend on which other variables are included in the model (through the off-diagonal terms in $(\mathbf{x}^T\mathbf{x})^{-1}$), this is a *very* specific hypothesis. In particular, whether the null hypothesis that $\beta_i = 0$ is true or not can easily depend on what other variables are included in the regression. What is really being checked here is, in ordinary language, something like "If you included all these other variables, would the model really fit *that* much better if you gave X_i a non-zero slope?"

13.3.3.1 Why, on Earth, Would You Want to Test That?

I am afraid that usually the answer is "you do not actually want to test that". You should ask yourself, carefully, whether it would really make any difference to you to know that the coefficient was precisely zero. (See Chapter 7, for some ideas about when that's worth testing and when it isn't.)

13.3.3.2 What Will Tend to Make a $\hat{\beta}$ Significant?

The t statistic for testing $\beta_i = 0$ is

$$\frac{\hat{\beta}_i}{\widehat{\mathrm{se}}\left[\hat{\beta}_i\right]} \tag{13.42}$$

We know that $\hat{\beta}_i$, being unbiased, will have a distribution centered on β_i, and the typical deviation away from that will in fact be about $\widehat{\mathrm{se}}\left[\hat{\beta}_i\right]$ in size, so we need to get a grip on that standard error.

From the theory above,

$$\widehat{\mathrm{se}}\left[\hat{\beta}_i\right] = \sqrt{\frac{\hat{\sigma}^2}{n}\left(\frac{1}{n}\mathbf{x}^T\mathbf{x}\right)^{-1}} \tag{13.43}$$

You showed in the homework (problem 3) that

$$\left(\frac{1}{n}\mathbf{x}^T\mathbf{x}\right) = \left[\begin{array}{ccccc} 1 & \overline{x_1} & \overline{x_2} & \ldots & \overline{x_p} \\ \overline{x_1} & \overline{x_1^2} & \overline{x_1 x_2} & \ldots & \overline{x_1 x_p} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \overline{x_p} & \overline{x_1 x_p} & \ldots & \overline{x_p^2} & \end{array}\right] \tag{13.44}$$

What will happen when we invert this? You can check (Exercise 4) that if $\overline{x_i x_j} = \bar{x_i}\bar{x_j}$ for all i, j, we'll get a diagonal matrix. Except for the very first entry on the diagonal (corresponding to the intercept), the entries will be inversely proportional to the variances of the predictor variables. If $\overline{x_i x_j} \neq \bar{x_i}\bar{x_j}$, the predictors are correlated, and this is going to increase the variance of their coefficients.

So, to sum up, four things control the standard error in $\hat{\beta}_i$: σ^2, the variance around the true regression function, since all standard errors are proportional to σ; n, since (all else being equal) all the standard errors are proportional to $1/\sqrt{n}$; the sample variance of X_i (since having data more widely spread on that axis makes it easier to find the slope); and the sample correlation between X_i and the other X_j (since strong correlations, positive or negative, make it harder to find their *specific* slopes).

What are the consequences?

1. Since, on any one data set, σ^2 and n are the same for all coefficients, the ones which are going to have the biggest test statistics, and so be "most significant", are the ones where (i) $|\beta_i|$ is large, (ii) the sample variance of X_i is large, and (iii) the sample correlation of X_i with other predictors is small.

2. The coefficients with the smallest p-values aren't necessarily the largest, let alone the most important; they may just be the most precisely measured.

3. Two people dealing with the same system, with precisely the same parameters and even the same n, can find different sets of coefficients to be significant, if their design matrices $\mathbf{x}$ differ. In fact, there need be no overlap in which coefficients are significant at all[1].

4. Adding or removing predictors will change which coefficients are significant, not just by changing the β_i, but also changing the standard error.

[1]In this case, the natural thing to do would be to combine the data sets.

5. Holding all the parameters fixed and letting n grow, the t statistic will go off to $\pm\infty$, unless $\beta_i = 0$ exactly. Every non-zero coefficient eventually becomes significant at arbitrarily small levels.

The same reasoning as in Chapter 7 shows that p-values will tend to go to zero exponentially fast as n grows, unless of course $\beta_i = 0$.

13.3.3.3 Things It Would Be Very Stupid to Do, So Of Course You Would Never Even *Think* of Doing Them

- Saying "β_i wasn't significantly different from zero, so X_i doesn't matter for Y". After all, X_i could still be an important cause of Y, but we don't have enough data, or enough variance on X_i, or enough variance in X_i uncorrelated with other X's, to accurately estimate its slope. All of these would prevent us from saying that β_i was *significantly* different from 0, i.e., distinguishable from 0 with high reliability.

- Saying "β_i was significantly different from zero, so X_i really matters to Y". After all, any β_i which is not *exactly* zero can be made arbitrarily significant by increasing n and/or the sample variance of X_i. That is, its t statistic will go to $\pm\infty$, and the p-value as small as you have patience to make it.

- Deleting all the variables whose coefficients didn't have stars by them, and re-running the regression. After all, since it makes no sense to pretend that the statistically significant variables are the only ones which matter, limiting the regression to the statistically significant variables is even less sensible.

- Saying "all my coefficients are really significant, so the linear-Gaussian model must be right". After all, all the hypothesis tests on individual coefficients *presume* the linear Gaussian model, both in the null and in the alternative. The tests have no power to notice nonlinearities, non-constant noise variance, or non-Gaussian noise.

13.4 Further Reading

The marginal figures are taken from Allie Brosh, "This Is Why I'll Never Be an Adult", *Hyperbole and a Half*, 17 June 2010, without permission but with the deepest possible respect. If these notes do nothing beyond inspiring you to read one of the greatest moral psychologists of our age, they will have done more than many classes.

On a profoundly lower plane, Berk (2004) has one of the most sensible discussions of the uses and abuses of statistical inference in multiple regression I know of.

For an extensive discussion of additive models, where we automatically search for transformations of the predictors, see Shalizi (forthcoming, ch. 9). Single-index models are used widely in econometrics; see, for instance, Li and Racine (2007).

13.5 Exercises

1. Prove Eq. 13.10.
2. Prove Eq. 13.23
3. *What if all null hypotheses were true?* (After Freedman 1983) Draw a $\mathbf{Y}$ from a standard Gaussian distribution with 1000 observations. Draw $\mathbf{X}$ by setting $p = 100$, and giving each X_i a standard Gaussian distribution.
   (a) Regress Y on all 100 X's (plus an intercept). How many of the β_is are significant at the 10% level? At the 5% level? At the 1% level? What is the R^2? The adjusted R^2?
   (b) Re-run the regression using just the variables which are significant at the 5% level. Plot a histogram of the change in coefficient for each variable from the old regression to the new regression. How many variables are now significant at the 1% level? What is the R^2? The adjusted R^2?
4. *Standard errors and correlations among the predictors* Assume that $p = 2$, so $n^{-1}\mathbf{x}^T\mathbf{x}$ is a 3×3 matrix.
   (a) Suppose that $\overline{x_1 x_2} = \bar{x_1}\bar{x_2}$, so there is no sample covariance between the two predictors. Find $(\frac{1}{n}\mathbf{x}^T\mathbf{x})^{-1}$ in terms of $\bar{x_1}$, $\bar{x_2}$, $\overline{x_1^2}$ and $\overline{x_2^2}$. Simplify, where possible, to eliminate second moments in favor of variances.
   (b) Give the general form of the inverse, $(\frac{1}{n}\mathbf{x}^T\mathbf{x})^{-1}$, without assuming $\overline{x_1 x_2} = \bar{x_1}\bar{x_2}$. How, qualitatively, do the variances of the slope estimates depend on the variances and covariances of the predictors?

Chapter 14

Polynomial and Categorical Regression

14.1 Essentials of Multiple Linear Regression

We predict a scalar random variable Y as a linear function of p different predictor variables $X_1, \ldots X_p$, plus noise:

$$Y = \beta_0 + \beta_1 X_1 + \ldots \beta_p X_p + \epsilon$$

and assume that $\mathbb{E}[\epsilon|X] = 0$, $\mathrm{Var}[\epsilon|X] = \sigma^2$, with ϵ being uncorrelated across observations. In matrix form,

$$\mathbf{Y} = \mathbf{X}\beta + \epsilon$$

the design matrix $\mathbf{X}$ including an extra column of 1s to handle the intercept, and $\mathbb{E}[\epsilon|\mathbf{X}] = 0$, $\mathrm{Var}[\epsilon|\mathbf{X}] = \sigma^2\mathbf{I}$.

If we add the Gaussian noise assumption, $\epsilon \sim MVN(0, \sigma^2\mathbf{I})$, independent of all the predictor variables.

The least squares estimate of the coefficient vector, which is also the maximum likelihood estimate if the noise is Gaussian, is

$$\widehat{\beta} = (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{y}$$

These are unbiased, with variance $\sigma^2(\mathbf{x}^T\mathbf{x})^{-1}$. Under the Gaussian noise assumption, $\widehat{\beta}$ itself has a Gaussian distribution. The standard error $\widehat{\mathrm{se}}\left[\hat{\beta}_i\right] = \sigma\sqrt{(\mathbf{x}^T\mathbf{x})^{-1}_{ii}}$. Fitted values are given by $\mathbf{x}\widehat{\beta} = \mathbf{H}\mathbf{y}$, and residuals by $\mathbf{e} = (\mathbf{I} - \mathbf{H})\mathbf{y}$. Fitted values $\mathbf{bm}$ and residuals $\mathbf{e}$ are also unbiased and have Gaussian distributions, with variance matrices $\sigma^2\mathbf{H}$ and $\sigma^2(\mathbf{I} - \mathbf{H})$, respectively.

When (as is usually the case) σ^2 is unknown, the maximum likelihood estimator is the in-sample mean-squared error, $n^{-1}(\mathbf{e}^T\mathbf{e})$ is a negatively biased estimator of σ^2: $\mathbb{E}\left[\hat{\sigma}^2\right] = \sigma^2\frac{n-p-1}{n}$. Under the Gaussian noise assumption, $n\hat{\sigma}^2/\sigma^2 \sim \chi^2_{n-p-1}$. Also under the Gaussian noise assumption, the Gaussian sampling distribution of any particular coefficient or conditional mean can be converted into a t distribution, with $n-p-1$ degrees of freedom, by using the appropriate standard error, obtained by plugging in the de-biased estimate of σ^2.

None of these results require any assumptions on the predictor variables X_i, except that they take real numerical values, and that they are linearly independent.

14.2 Adding Curvature: Polynomial Regression

Because the predictor variables are almost totally arbitrary, there is no harm in making one predictor variable a function of another, so long as it isn't a linear function. In particular, there is nothing wrong with a model like

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_1^2 + \ldots \beta_d X_1^d + \beta_{d+1} X_2 + \ldots \beta_{p+d-1} X_p + \epsilon$$

where instead of Y being linearly related to X_1, it's polynomially related, with the order of the polynomial being d. We just add $d-1$ columns to the design matrix $\mathbf{x}$, containing $x_1^2, x_1^3, \ldots x_1^d$, and treat them just as we would any other predictor variables. With this expanded design matrix, it's still true that $\widehat{\beta} = (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{y}$, that fitted values are $\mathbf{Hy}$ (using the expanded $\mathbf{x}$ to get $\mathbf{H}$), etc. The number of degrees of freedom for the residuals will be $n-(p+1+(d-1))$.

Nor is there principled reason why every predictor variable can't have its own polynomial, each with (potentially) a different degree d_i. In that case, numbering the βs sequentially gets tricky, and better notation would be something like

$$Y = \beta_0 + \sum_{i=1}^{p}\sum_{j=1}^{d_i} \beta_{i,j} X_i^j + \epsilon$$

though then we'd have to remember to "stack" the $\beta_{i,j}$s into a vector of length $1 + \sum_{i=1}^{p} d_i$ for estimation.

Mathematically, we are treating X_i and X_i^2 (and X_i^3, etc.) as distinct predictor variables, but that's fine, since they won't be linearly dependent on each other[1], or linearly dependent on other predictors[2]. Again, we just expand the design matrix with extra columns for all the desired powers of each predictor variable. The number of degrees of freedom for the residuals will be $n-(1+\sum_i d_i)$.

There are a bunch of mathematical and statistical points to make about polynomial regression, but let's take a look at how we'd actually estimate one of these models in R first.

[1]Well, hardly ever: if X_i was only ever, say, 0 or 1, then it would be equal to X_i^2. Such awkward cases happen with probability 0 for continuous variables.

[2]Again, you can contrive awkward cases where this is not true, if you really want to. For instance, if X_1 and X_2 are horizontal and vertical coordinates of points laid out on a circle, they are linearly independent of each other and of their own squares, but X_1^2 and X_2^2 are linearly dependent. (Why?) The linear dependence would be broken if the points were laid out in an ellipse or oval, however. (Why?)

14.2.1 R Practicalities

There are a couple of ways of doing polynomial regression in R.

The most basic is to manually add columns to the data frame with the desired powers, and then include those extra columns in the regression formula:

```
df$x.sq <- df$x^2
lm(y ~ x + x.sq, data = df)
```

I do not recommend using this form, since it means that you need to do a lot of repetitive, boring, error-prone work, and get it exactly right. (For example, to do predictions with `predict`, you'd need to specify the values for all the powers of all the predictors.)

A somewhat more elegant alternative is to tell R to use various powers in the formula itself:

```
lm(y ~ x + I(x^2), data = df)
```

Here `I()` is the **identity function**, which tells R "leave this alone". We use it here because the usual symbol for raising to a power, `^`, has a special meaning in linear-model formulas, relating to interactions. (We'll cover this in Chapter 17, or, if you're impatient, see `help(formula.lm)`.) When you do this, `lm` will create the relevant columns in the matrix it uses internally to calculate the estimates, but it leaves `df` alone. When it comes time to make a prediction, however, R will take care of the transformations on the new data.

Finally, since it can grow tedious to write out all the powers one wants, there is the convenience function `poly`, which will create all the necessary columns for a polynomial of a specified degree:

```
lm(y ~ poly(x, 2), data = df)
```

Here the second argument, `degree`, tells `poly` what order of polynomial to use. R remembers how this works when the estimated model is used in `predict`. My *advice* is to use `poly`, but the other forms aren't wrong.

Small demo Here is a small demo of polynomial regression, using the data from the first data analysis project.

```
# Load the data
mobility <- read.csv("http://www.stat.cmu.edu/~cshalizi/mreg/15/dap/1/mobility.csv")
mob.quad <- lm(Mobility ~ Commute + poly(Latitude, 2) + Longitude, data = mobility)
```

This fits a quadratic in the `Latitude` variable, but linear terms for the other two predictors. You will notice that `summary` does nothing strange here:

```
summary(mob.quad)
##
## Call:
## lm(formula = Mobility ~ Commute + poly(Latitude, 2) + Longitude,
##     data = mobility)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.12828 -0.02384 -0.00691  0.01722  0.32190
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        -0.0261223  0.0121233  -2.155   0.0315
## Commute             0.1898429  0.0137167  13.840  < 2e-16
## poly(Latitude, 2)1  0.1209235  0.0475524   2.543   0.0112
## poly(Latitude, 2)2 -0.2596006  0.0484131  -5.362 1.11e-07
## Longitude          -0.0004245  0.0001394  -3.046   0.0024
##
## Residual standard error: 0.04148 on 724 degrees of freedom
## Multiple R-squared:  0.3828,Adjusted R-squared:  0.3794
## F-statistic: 112.3 on 4 and 724 DF,  p-value: < 2.2e-16
```

and we can use `predict` as usual:

```
predict(mob.quad, newdata = data.frame(Commute = 0.298, Latitude = 40.57, Longitude = -79.58))
##          1
## 0.07079416
```

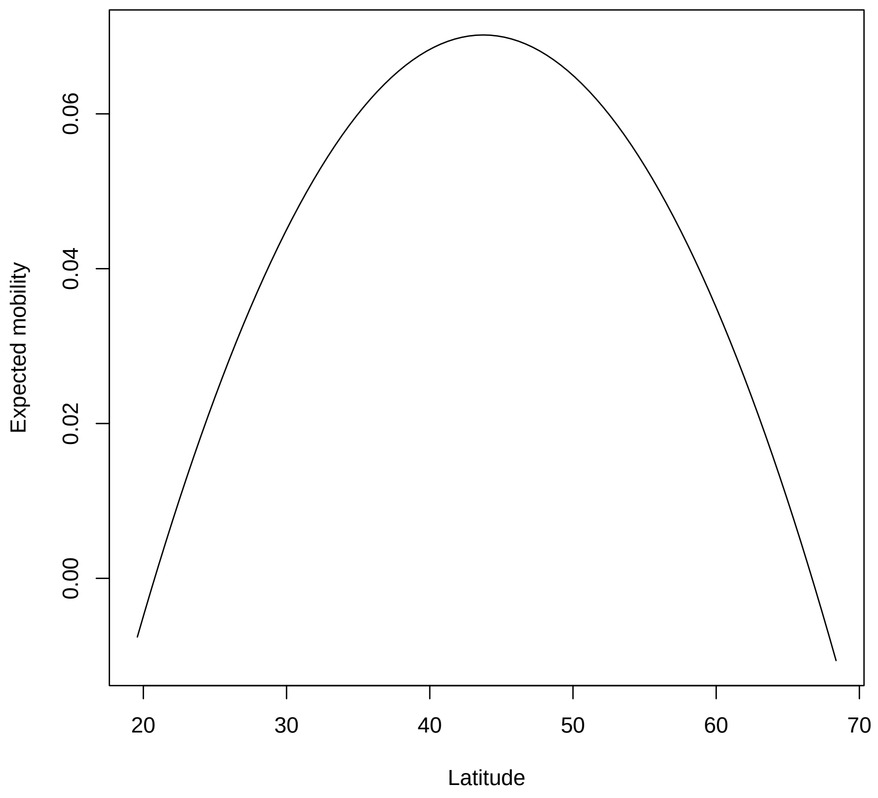
See also Figure 14.1 for an illustration that this really is giving us behavior which is non-linear in the `Latitude` variable.

14.2.2 Properties, Issues, and Caveats

Diagnostic plots The appropriate diagnostic plot is of residuals against the predictor. There is no need to make separate plots of residuals against each power of the predictor.

Smoothness Polynomial functions vary continuously in all their arguments. In fact, they are "smooth" in the sense in which mathematicians use that word, meaning that all their derivatives exist and are continuous, too. This is desirable if you think the real regression function you're trying to model is smooth, but not if you think there are sharp thresholds or jumps. Polynomials *can* approximate thresholds arbitrarily closely, but you end up needing a very high order polynomial.

Interpretation In a linear model, we were able to offer simple interpretations of the coefficients, in terms of slopes of the regression surface.



```
hypothetical.pghs <- data.frame(Commute = 0.287, Latitude = seq(from = min(mobility$Latitude),
    to = max(mobility$Latitude), length.out = 100), Longitude = -79.92)
plot(hypothetical.pghs$Latitude, predict(mob.quad, newdata = hypothetical.pghs),
    xlab = "Latitude", ylab = "Expected mobility", type = "l")
```

FIGURE 14.1: *Predicted rates of economic mobility for hypothetical communities at the same longitude as Pittsburgh, and with the same proportion of workers with short commutes, but different latitudes.*

In the multiple linear regression model, we could say

$$\beta_i = \mathbb{E}\left[Y|X_i = x_i + 1, X_{-i} = x_{-i}\right] - \mathbb{E}\left[Y|X_i = x_i, X_{-i} = x_{-i}\right]$$

("β_i is the difference in the expected response when X_i is increased by one unit, all other predictor variables being equal"), or

$$\beta_i = \frac{\mathbb{E}\left[Y|X_i = x_i + h, X_{-i} = x_{-i}\right] - \mathbb{E}\left[Y|X_i = x_i, X_{-i} = x_{-i}\right]}{h}$$

("β_i is the slope of the expected response as X_i is varied, all other predictor variables being equal"), or

$$\beta_i = \frac{\partial \mathbb{E}\left[Y|X = x\right]}{\partial x_i}$$

("β_i is the rate of change in the expected response as X_i varies"). None of these statements is true any more in a polynomial regression.

Take them in reverse order. The rate of change in $\mathbb{E}\left[Y|X\right]$ when we vary X_i is now

$$\frac{\partial \mathbb{E}\left[Y|X = x\right]}{\partial x_i} = \sum_{j=1}^{d} j \beta_{i,j} x_i^{j-1}$$

This not only involves all the coefficients for all the powers of X_i, but also has a different answer at different points x_i. The linear coefficient on X_i, $\beta_{i,1}$, is the rate of change when $X_i = 0$, but not otherwise. There just is no one answer to "what's the rate of change?".

Similarly, if we ask for the slope,

$$\frac{\mathbb{E}\left[Y|X_i = x_i + h, X_{-i} = x_{-i}\right] - \mathbb{E}\left[Y|X_i = x_i, X_{-i} = x_{-i}\right]}{h}$$

that isn't given by one single number either; it depends on the starting value x_i and the size of the change h. If h is very close to very, the slope will be approximately $h\sum_{j=1}^{d} j \beta_{i,j} x_i^{j-1}$, but not, generally, otherwise. If you really want to know, you have to actually plug in to the polynomial.

Finally, the change associated with a one-unit change in X_i is just a special case of the slope when $h = 1$, and so not equal to any of the coefficients either. It will definitely change as the starting point x_i changes.

Rather than trying to give one single rate of change (or slope or response-associated-to-a-one-unit-change) when none exists, a more honest procedure is to make a plot, either of the polynomial itself, or of the derivative. (See the example in the model report for the first DAP.)

Interpreting the polynomial as a transformation of X_i If you really wanted to, you could try to complete the square (cube, other polynomial) to re-write the polynomial

$$\beta_1 X_1 + \beta_2 X_1^2 + \ldots \beta_d X_1^d = k + \beta_d \prod_{j=1}^{d} (X_1 - c_j)$$

You could then say that β_d was the change in the response for a one-unit change in $\prod_{j=1}^{d}(X_1 - c_j)$, etc., etc. The zeroes or roots of the polynomial, c_j, will be functions of the coefficients on the lower powers of X_1, but their sampling distributions, unlike those of the β_j, would be very tricky, and so, consequently, would their confidence sets. Moreover, it is not very common for the transformed predictor $\prod_{j=1}^{d}(X_1 - c_j)$ to itself be a particularly interpretable variable, so this is often a considerable amount of work for little gain.

"Testing for nonlinearity" It is not uncommon to see people claiming to test whether the relationship between Y and X_i is linear by adding a quadratic term in X_i and testing whether the coefficient on it significantly different from zero. This would work fine if you knew that the only possible sort of nonlinearity was quadratic — that if the relationship wasn't a straight line, it was a parabola. Since it is perfectly possible to have a very nonlinear relationship where the coefficient on X_i^2 is zero, this is not a very powerful test.

Over-fitting and wiggliness A polynomial of degree d can exactly fit any d points. (Any two points lie on a line, any three on a parabola, etc.) Using a high-order polynomial, or even summing a large number of low-order polynomials, can therefore lead to curves which come very close to the data we used to estimate them, but predict very badly. In particular, high-order polynomials can display very wild oscillations in between the data points. Plotting the function in between the data points (using `predict`) is a good way of noting this. We will also look at more formal checks when we cover cross-validation later in the course.

Picking the polynomial order The best way to pick the polynomial order is on the basis of some actual scientific theory which says that the relationship between Y and X_i should, indeed, by a polynomial of order d_i. Failing that, carefully examining the diagnostic plots is your next best bet. Finally, the methods we'll talk about for variable and model selection in forthcoming chapters can also be applied to picking the order of a polynomial, though as we will see, you need to be very careful about what those methods actually do, and whether that's really what you want.

14.2.3 Orthogonal Polynomials

I have written out polynomial regression above in its most readily-comprehended way, but that is not always the most best way to estimate it. We know, from our previous examination of multiple linear regression, that we'll get smaller standard errors when our predictor variables are uncorrelated. While X_i and its higher powers are linearly independent, they are generally (for most distributions) somewhat correlated. An alternative to regressing on the powers of X_i is to regress on linear function of X_i, a quadratic function of X_i, a cubic, etc., which are chosen so that they are *un*-correlated on the data. These functions, being uncorrelated, are called **orthogonal**. Any polynomial could also be expressed as a linear combination of these **basis functions**, which are thus called **orthogonal polynomials**. The advantage, again, is that the estimates of coefficients on these basis functions have less variance than using the powers of X_i.

In fact, this is what the `poly` function does by default; to force it to use the powers of X_i, we need to set the `raw` option to `TRUE`.

To be concrete, let's start with the linear function. We'll arrange it so that it has mean zero (and therefore doesn't contribute to the intercept):

$$\sum_{i=1}^{n} \alpha_{i10} + \alpha_{i11} x_{i1} = 0$$

Here I am using α_{ijk} to indicate the coefficient on X_i^k in the j^{th} order basis function for X_i. This is one equation with two unknowns, so we need another equation to be able to solve the system. What `poly` does is to impose a constraint on the sample variance:

$$\sum_{i=1}^{n} (\alpha_{i10} + \alpha_{i11} x_{i1})^2 = 1$$

(Why is this a constraint on the variance?) The quadratic function is found by requiring that it have mean zero,

$$\sum_{i=1}^{n} \alpha_{i20} + \alpha_{i21} x_{i1} + \alpha_{i22} x_{i1}^2 = 0 \, ,$$

that it be uncorrelated with the linear function,

$$\sum_{i=1}^{n} (\alpha_{i10} + \alpha_{i11} x_{i1}) \left(\alpha_{i20} + \alpha_{i21} x_{i1} + \alpha_{i22} x_{i1}^2\right) = 0 \, ,$$

and that it have the same variance as the linear function:

$$\sum_{i=1}^{n} \left(\alpha_{i20} + \alpha_{i21} x_{i1} + \alpha_{i22} x_{i1}^2\right)^2 = 1$$

To get the j^{th} basis function, we need all the $j-1$ basis functions that came before it, so we can make sure it has mean 0, that it's uncorrelated with all of the others, and that it has the same variance. All of the coefficients I've written α are encoded in the attributes of the output of `poly`, though not always in an especially humanly-readable way. (For details, see `help(poly)`, and the references it cites.)

Notice that changing the sample values of X_i will change the basis functions; one reason to use the powers of X_i instead would be to make it easier to compare coefficients across data sets. If the distribution of X_i is known, one can work out systems of orthogonal polynomials in advance, for instance, the Legendre polynomials which are orthogonal when the predictor variable has a uniform distribution[3].

[3]See, for instance, Wikipedia, s.v. "Legendre polynomials".

14.2.4 Non-Polynomial Function Bases

There are basically three reasons to want to use polynomials. First, many scientific theories claim that there are polynomial relationships between variables in the real world. Second, they're things we've all been familiar with since basic algebra, so we understand them very well, we find them un-intimidating, and very little math is required to use them. Third, they have the nice property that any well-behaved function can be approximated arbitrarily closely by a polynomial of sufficiently high degree[4].

If we don't have strong scientific reasons to want to use polynomials, and are willing to go beyond basic algebra, there are many other systems of functions which also have the universal approximation property. If we're just doing curve fitting, it can be just as good, and sometimes much better, to use one of these other function bases. For instance, we might use sines and cosines at multiples of a basic frequency ω,

$$\sum_{j=1}^{d} \gamma_{i1j} \sin(j\omega X_i) + \gamma_{i2j} \cos(j\omega X_i)$$

Such a basis would be especially appropriate for variables which are really angles, or when there is a periodicity in the system. Exactly matching a sum of sines and cosines like the above would require an infinite-order polynomial; conversely, matching a linear function with a sum of sines and cosines would require letting $d \to \infty$.

As this suggests, there is a bit of an art to picking a suitable function basis; as it also suggests, it's an area where knowledge of more advanced mathematics (specifically, functional analysis) can be really useful to actually doing statistics.

[4]See further reading, below, for details.

14.3 Categorical Predictors

We often have variables which we think are related to Y which are not real numbers, but are qualitative rather than quantitative — answers to "what kind?" rather than to "how much?". For people, these might be things like sex, gender, race, caste, religious affiliation, education attainment, occupation, whether they've had chicken pox, whether they have previously defaulted on a loan, or their country of citizenship. For geographic communities (as in the data analysis project), state was a categorical variable, though not one we used because we didn't know how.

Some of these are purely qualitative, coming in distinct types, but with no sort of order or ranking implied; these are often specifically called "categorical", and the distinct values "categories". (The values are also called "levels", though that's not a good metaphor without an order.) Other have distinct levels which can be put in a sensible order, but there is no real sense that the *distance* between one level and the next is the same — they are **ordinal** but not **metric**. When it is necessary to distinguish non-ordinal categorical variables, they are often called **nominal**, to indicate that their values have names but no order.

In R, categorical variables are represented by a special data type called `factor`, which has as a sub-type for ordinal variables the data type `ordered`.

In this section, we'll see how to include both categorical and ordinal variables in multiple linear regression models, by **coding** them as numerical variables, which we know how to handle.

14.3.1 Binary Categories

The simplest case is that of a binary variable B, one which comes in two qualitatively different types. To represent this in a format which fits with the regression model, we pick one of the two levels or categories as the "reference" or "baseline" category. We then add a column X_B to the design matrix $\mathbf{x}$ which indicates, for each data point, whether it belongs to the reference category ($X_B = 0$) or to the other category ($X_B = 1$). This is called an **indicator variable** or **dummy variable**. That is, we **code** the qualitative categories as 0 and 1.

We then regress on the indicator variable, along with all of the others, getting the model

$$Y = \beta_0 + \beta_B X_b + \beta_1 X_1 + \ldots \beta_p X_p + \epsilon$$

The coefficient β_b is the expected difference in Y between two units which are identical, except that one of them has $X_b = 0$ and the other has $X_b = 1$. That is, it's the expected difference in the response between members of the reference category and members of the other category, all else being equal. For this reason, β_B is often called the **contrast** between the two classes.

Geometrically, if we plot the expected value of Y against $X_1, \ldots X_p$, we will now get *two* regression surfaces: they will be parallel to each other, and offset by β_B. We thus have a model where each category gets its own intercept: β_0 for the reference class, $\beta_0 + \beta_B$ for the other class. You should, at this point, convince yourself that if we had switched which class was the reference class, we'd get exactly the same slopes, only with the over-all intercept being $\beta_0 + \beta_B$ and the contrast being $-\beta_B$ (Exercise 1).

In R If a data frame has a column which is a two-valued factor already, and it's included in the right-hand side of the regression formula, `lm` handles creating the column of indicator variables internally.

Here, for instance, we use a classic data set to regress the weight of a cat's heart on its body weight and its sex. (If it worked, such a model would be useful in gauging doses of veterinary heart medicines.)

```
library(MASS)
data(cats)
Hwt.lm <- lm(Hwt ~ Sex + Bwt, data = cats)
summary(Hwt.lm)
##
## Call:
## lm(formula = Hwt ~ Sex + Bwt, data = cats)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.5833 -0.9700 -0.0948  1.0432  5.1016
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.4149     0.7273  -0.571    0.569
## SexM         -0.0821     0.3040  -0.270    0.788
## Bwt           4.0758     0.2948  13.826   <2e-16
##
## Residual standard error: 1.457 on 141 degrees of freedom
## Multiple R-squared:  0.6468,Adjusted R-squared:  0.6418
## F-statistic: 129.1 on 2 and 141 DF,  p-value: < 2.2e-16
```

`Sex` is coded as `F` and `M`, and R's output indicates that it chose `F` as the reference category.

Diagnostics The mean of the residuals within each category is guaranteed to be zero (Exercise 2), but they should also have the same variance and otherwise the same distribution, so there is still some point in plotting residuals against X_B. Sometimes a little jitter on the horizontal axis helps, or making a box-plot.

Inference There is absolutely nothing special about the inferential statistics for the estimated contrast $\hat{\beta}_B$. It works just like inference for any other regression coefficient.

Why not just split the data? If we want to give each class its own intercept, why not just split the data and estimate two models, one for each class? The answer is that sometimes we'll do just this, especially if there's a lot of data for each class. However, if the regression surfaces for the two categories really are parallel to each other, by splitting the data we're losing some precision in our estimate of the common slopes, without gaining anything. In fact, if the two surfaces are *nearly* parallel, for moderate sample sizes the small bias that comes from pretending the slopes are all equal can be overwhelmed by the reduction in variance.

Why not two columns? It's natural to wonder why we have to pick out one level as the reference, and estimate a contrast. Why not add *two* columns to $\mathbf{x}$, one indicating each class? The problem is that then those two columns will be linearly dependent (they'll always add up to one), so the data would be collinear and the model in-estimable.

Why not two slopes? The model we've specified has two parallel regression surfaces, with the same slopes but different intercepts. We could also have a model with the same intercept across categories, but different slopes for each variable. Geometrically, this would mean that the regression surfaces weren't parallel, but would meet at the origin (and elsewhere). We'll see how to make that work when we deal with interactions in Chapter 17. If we wanted different slopes and intercepts, we might as well just split the data.

Contrasts need contrasts Just as we can't estimate β_i if $\mathrm{Var}[X_i] = 0$, we can't estimate any categorical contrasts if all the data points belong to the same category.

14.3.1.1 "Adjusted effect of a category"

As I said, β_B is the expected difference in Y between two individuals which have the same value for all of the variables *except* the category. This is generally *not* the same as the difference in expectations between the two categories:

$$\beta_B \neq \mathbb{E}[Y|X_B = 1] - \mathbb{E}[Y|X_B = 0]$$

One of the few situations where $\beta_B = \mathbb{E}[Y|X_B = 1] - \mathbb{E}[Y|X_B = 0]$ is when the distribution of all the *other* variables is the same between the categories. (Said another way, the categories are statistically independent of the other predictors.) Another is when there are no other predictors.

Because of this, it's very natural to want to interpret β_B as the difference in the response between the two groups, *adjusting for* all of the other variables. It's even common to talk about β_B as "the adjusted effect" of the category. As you might imagine, such interpretations come up all the time in disputes about discrimination.

Even leaving aside the emotional charge of such arguments, it is wise to be cautious about such interpretations, for several reasons.

1. The regression is only properly adjusting for all of the other variables if it's well-specified. If it's not, the contrast between the categories will also pick up some of the average difference in bias (due to getting the model wrong), which is not relevant.

2. As usual, finding that the contrast coefficient isn't significant doesn't necessarily mean there is no contrast! It means that the contrast, if there is one, can't be reliably distinguished from 0, which could be because it's very small or because we can't estimate it well. Again as usual, a confidence interval is called for.
3. It's not clear that we always *do* want to adjust for other variables, even when we can measure them. For instance, if economists in Lilliput found no effect on income between those who broke their eggs at the big end and those at the little end, after adjusting for education and occupational prestige (Swift, 1726), that wouldn't necessarily settle the question of whether big-endians were discriminated against. After all, it might be that they have less access to education and high-paid jobs *because* they were big-endians. And this could be true even if Lilliputians were initially randomly assigned between big- and little- end-breaking. The same goes for finding that there *is* an "adjusted effect".

The last point brings us close to topics of causal inference, which we won't get to until 402. For now, a good rule of thumb is not to adjust for variables which might themselves be effects of the variable we're interested in.

14.3.2 Categorical Variables with More than Two Levels

Suppose our categorical variable C has more than two levels, say k of them. We can handle it in almost exactly the same way as the binary case. We pick one level — it really doesn't matter which — as the reference level. We then introduce $k-1$ columns into the design matrix $\mathbf{x}$, which are indicators for the other categories. If, for instance, $k=3$ and the classes are `North`, `South`, `West`, we pick one level, say `North`, as the reference, and then add a column $X_{\mathtt{South}}$ which is 1 for data points in class `South` and 0 otherwise, and another column $X_{\mathtt{West}}$ which is 1 for data points in that class and 0 otherwise.

Having added these columns to the design matrix, we regress as usual, and get $k-1$ contrasts. The over-all β_0 is really the intercept for the reference class; the contrasts are added to β_0 to get the intercept for each class. Geometrically, we now have k parallel regression surfaces, one for each level of the variable.

Interpretation $\beta_{C=c}$ is the expected difference between two individuals who are otherwise identical, except that one is in the reference category and the other is in class c. The expected difference between two otherwise-identical individuals in two different categories, say c and d, is therefore the difference in their contrasts, $\beta_{C=d} - \beta_{C=c}$.

Diagnostics and inference Work just the same as in the binary case.

Why not k columns? Because, just like in the binary case, that would make all those columns for that variable sum to 1, causing problems with collinearity.

Contrasts need contrast If we know there are k categories, but some of them don't appear in our data, we can't estimate their contrasts.

Category-specific slopes and splitting the data The same remarks apply as under binary predictor variables.

14.3.3 Two, Three, Many Categorical Predictors

Nothing in what we did above requires that there be only one categorical predictor; the other variables in the model could be indicator variables for other categorical predictors. Nor do all the categorical predictors have to have the same number of categories. The only wrinkle with having multiple categories is that β_0, the over-all intercept, is now the intercept for individuals where *all* categorical variables are in their respective reference levels. Each combination of categories gets its own regression surface, still parallel to each other.

With multiple categories, it is natural to want to look at interactions — to let their be an intercept for left-handed little-endian plumber, rather than just adding up contrasts for being left-handed and being a little-endian and being a plumber. We'll look at that when we deal with interactions.

14.3.4 Analysis of Variance: Only Categorical Predictors

A model in which there are *only* categorical predictors is, for historical reasons, often called an **analysis of variance** model. Estimating such a model presents absolutely no special features beyond what we have already covered, but it's worth a paragraph or two on the interpretation and the origins of such models.

Suppose, for simplicity, that there are two categorical predictors, B and C, and the reference level for each is written $\emptyset$. The conditional expectation of Y will be pinned down by giving a level for each, say b and c, respectively. Then

$$\mathbb{E}[Y|B=b, C=c] = \beta_0 + \beta_b \delta_{b\emptyset} + \beta_c \delta_{c\emptyset}$$

That is, we add the appropriate contrast for each categorical variable, and nothing else. (This presumes no interactions, a limitation which we'll lift next week.) Conversely, if we knew $\mathbb{E}[Y|B=b, C=c]$ for every category, we could work out the contrasts without having to ever (explicitly) compute $(\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{y}$, which was a very real consideration before computation became so cheap[5]. Obviously, however, it is not much of an issue now.

As for the name, it arises from the basic probability fact sometimes called the "law of total variance":

$$\mathrm{Var}[Y] = \mathrm{Var}[\mathbb{E}[Y|X]] + \mathbb{E}[\mathrm{Var}[Y|X]]$$

If X is our complete set of categorical variables, each of which defines a group, this says "The total variance of the response is the variance in average responses across groups, plus the average variance within a group". Thus, after estimating the contrasts, we have decomposed or analyzed the variance in Y into between-group and across-group

[5]To see how, notice that $\hat{\beta}_0$ can be estimated by the sample mean of all cases where $B=\emptyset$, $C=\emptyset$. Then to get, say, β_b, we average the difference in means between cases where $B=b, C=c$ and $B=\emptyset, C=c$ for each level c of the other variable. (This averaging of differences eliminates the contribution from β_c.)

variance. This was extremely useful in the early days of agricultural and industrial experimentation, but has frankly become a bit of a fossil, if not a fetish.

An "analysis of covariance" model is just a regression with both qualitative and quantitative predictors.

14.3.5 Ordinal Variables

An ordinal variable, as I said, is one where the qualitatively-distinct levels can be put in a sensible order, but there's no implication that the distance from one level to the next is constant. At our present level of sophistication, we have basically two ways to handle them:

1. Ignoring the ordering and treat them like nominal categorical variables.
2. Ignoring the fact that they're only ordinal and not metric, assign them numerical codes (say 1, 2, 3, ...) and treat them like ordinary numerical variables.

The first procedure is unbiased, but can end up dealing with a lot of distinct coefficients. It also has the drawback that if the relationship between Y and the categorical variable is monotone, that may not be respected by the coefficients we estimate. The second procedure is very easy, but usually without any substantive or logical basis. It implies that each step up in the ordinal variable will predict exactly the *same* difference in Y, and why should that be the case? If, after treating an ordinal variable like a nominal one, we get contrasts which are all (approximately) equally spaced, we might then try the second approach.

Other procedures for ordinal variables which are, perhaps, more conceptually satisfying need much more math than we're presuming here; see the further reading.

14.3.6 Detailed R Example

The data set for the first data analysis project included a categorical variable, `State`, which we did not use. Let's try adding it to the model.

First, let's do some basic counting and examination:

```
# How many levels does State have?
nlevels(mobility$State)
## [1] 0
# What are they?
levels(mobility$State)
## NULL
```

There are 51 levels for `State`, as there should be, corresponding to the 50 states and the District of Columbia. We see that these are given by the two-letter postal codes, in alphabetical order.

Running a model with `State` and `Commute` as the predictors, we therefore expect to get 52 coefficients (1 intercept, 1 slope, and 51-1 = 50 contrasts). R will calculate contrasts from the first level, which here is `AK`, or Alaska.

```
mob.state <- lm(Mobility ~ Commute + State, data = mobility)
signif(coefficients(mob.state), 3)
## (Intercept)     Commute     StateAL     StateAR     StateAZ     StateCA
##    0.018400    0.126000   -0.005600    0.001840    0.007290    0.031500
##     StateCO     StateCT     StateDC     StateDE     StateFL     StateGA
##    0.044100    0.021500    0.071300    0.007460    0.004160   -0.022000
##     StateHI     StateIA     StateID     StateIL     StateIN     StateKS
##    0.029100    0.052200    0.029700    0.013200    0.010000    0.042400
##     StateKY     StateLA     StateMA     StateMD     StateME     StateMI
##    0.011900    0.021100    0.001230    0.018700    0.004710    0.005230
##     StateMN     StateMO     StateMS     StateMT     StateNC     StateND
##    0.055300    0.011900   -0.018700    0.045200   -0.011400    0.146000
##     StateNE     StateNH     StateNJ     StateNM     StateNV     StateNY
##    0.060400    0.032200    0.062700    0.006670    0.045400    0.022300
##     StateOH     StateOK     StateOR     StatePA     StateRI     StateSC
##   -0.000559    0.036000    0.013800    0.035000    0.022400   -0.019300
##     StateSD     StateTN     StateTX     StateUT     StateVA     StateVT
##    0.042300    0.000761    0.032200    0.060500    0.014100    0.017300
##     StateWA     StateWI     StateWV     StateWY
##    0.025800    0.031700    0.057800    0.061200
```

In the interest of space, I won't run `summary` on this, but you can. You will find that quite a few of the contrasts are statistically significant. We'd expect about $50 \times 0.05 = 2.5$ to be significant at the 5% level, even if all the true contrasts were zero, but many more them are than this baseline. As usual, of course, it doesn't mean the model is right; it just means that if we were going to put in an intercept, a slope for `Commute`, and a contrast for every other state, we should really put in contrasts for those states as well.

One issue with the simple linear regression from the DAP was that its residuals were very strongly correlated spatially. We might hope that adding all these state-by-state contrasts has gotten rid of some of that correlation.

When we have a large number of categories, it's often tempting to try compressing them to a smaller number, by grouping together some of the levels. If we do this right, we reduce the variance in our estimates of the coefficients, while introducing little (if any) bias.

To illustrate this, let's try boiling down the 51 states (and DC) into two categories: the South versus the rest of the country. The South has long been quite distinct from the rest of the country culturally, politically and economically, in ways which are, plausibly, very relevant to economic mobility. More relevantly, when we looked at the residuals in the DAP, there was a big cluster of negative residuals in the south-eastern part of the map. To make this concrete, I'll define the South as consisting of those states which joined the Confederacy during the Civil War (Alabama, Arkansas, Florida, Georgia, Louisiana, Mississippi, North Carolina, South Carolina, Tennessee, Texas and Virginia).

Let's start by adding the relevant column to the data frame:

```
# Set up a function to make maps 'Terrain' color levels set based on quantiles
# of the variable being plotted Inputs: vector to be mapped over the data
# frame; number of levels to use for colors; other plotting arguments Outputs:
# invisibly, list giving cut-points and the level each observation was assigned
mapper <- function(z, levels, ...) {
    # Which quantiles do we need?
    probs <- seq(from = 0, to = 1, length.out = (levels + 1))
    # What are those quantiles?
    z.quantiles <- quantile(z, probs)
    # Assign each observation to its quantile
    z.categories <- cut(z, z.quantiles, include.lowest = TRUE)
    # Make up a color scale
    shades <- terrain.colors(levels)
    plot(x = mobility$Longitude, y = mobility$Latitude, col = shades[z.categories],
        ...)
    invisible(list(quantiles = z.quantiles, categories = z.categories))
}
```

FIGURE 14.2: *Function for making maps, from the model DAP 1.*

```
# The states of the Confederacy
Confederacy <- c("AR", "AL", "FL", "GA", "LA", "MS", "NC", "SC", "TN", "TX", "VA")
mobility$Dixie <- mobility$State %in% Confederacy
```

The new `Dixie` column of `mobility` will contain the values `TRUE`, for each community located in one of those states, and `FALSE`, for the rest. R will in such circumstances treat `FALSE` as the reference category.

```
mob.dixie <- lm(Mobility ~ Commute + Dixie, data = mobility)
signif(coefficients(summary(mob.dixie)), 3)
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.0190    0.00607    3.13 1.84e-03
## Commute       0.1950    0.01180   16.50 2.94e-52
## DixieTRUE    -0.0217    0.00354   -6.14 1.37e-09
```

The contrast for the old Confederacy versus the rest of the country is negative, meaning those states have lower levels of economic mobility, and highly statistically significant. Of course, the model could still be wrong. The residuals, while better than a model with no geographic contrasts, don't look as random as in the one with contrasts for each state.

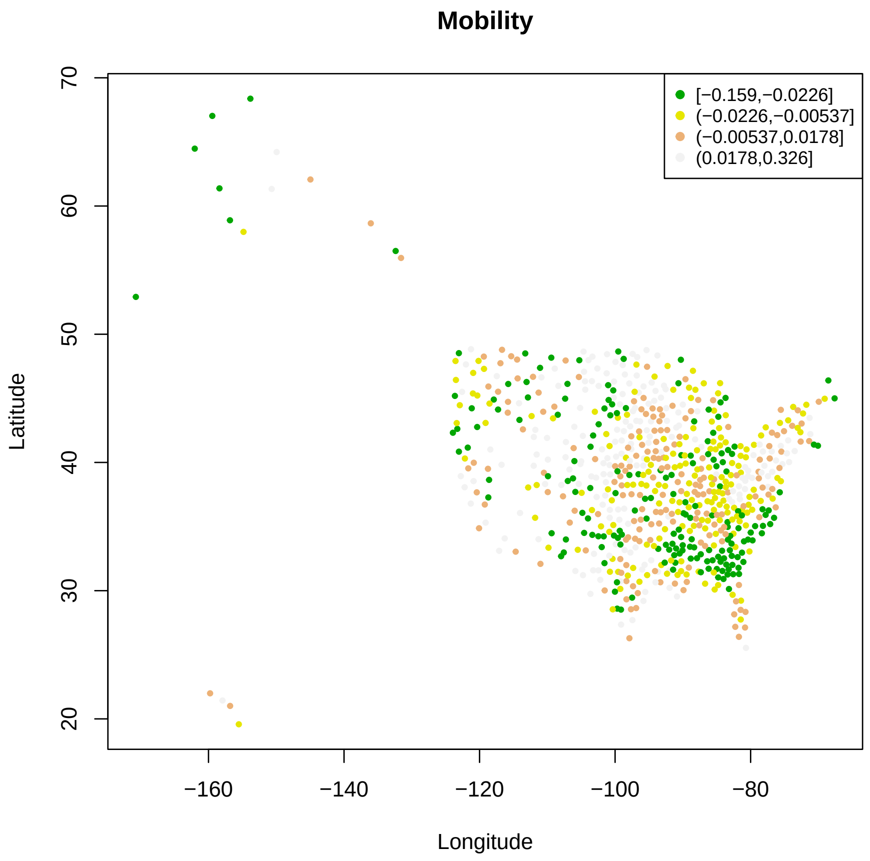
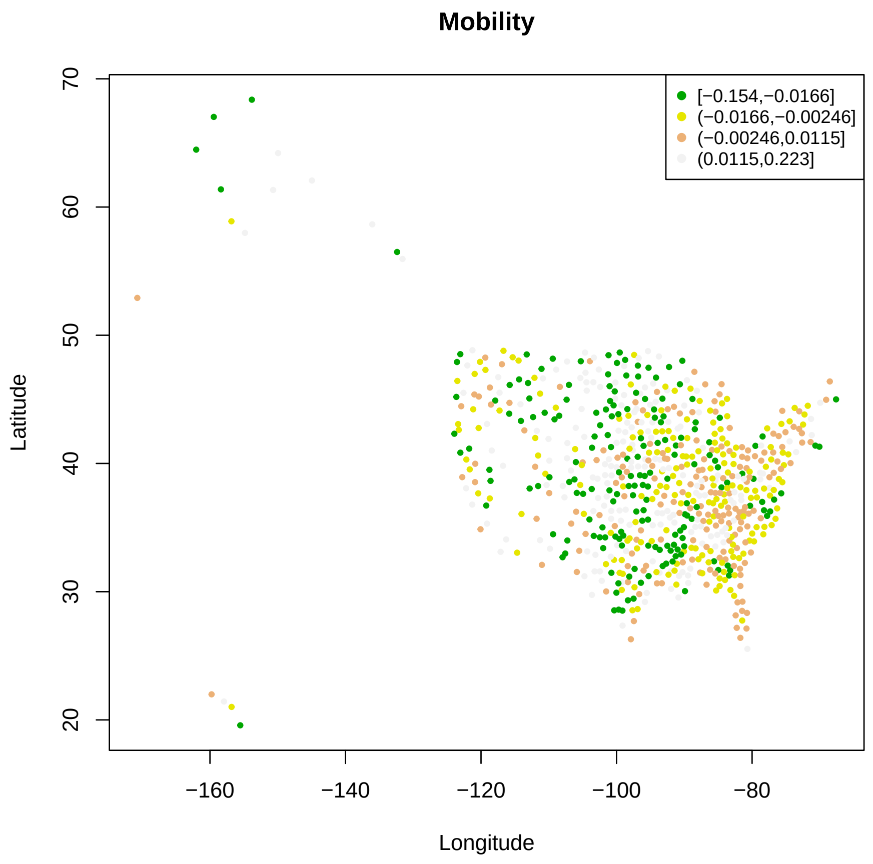
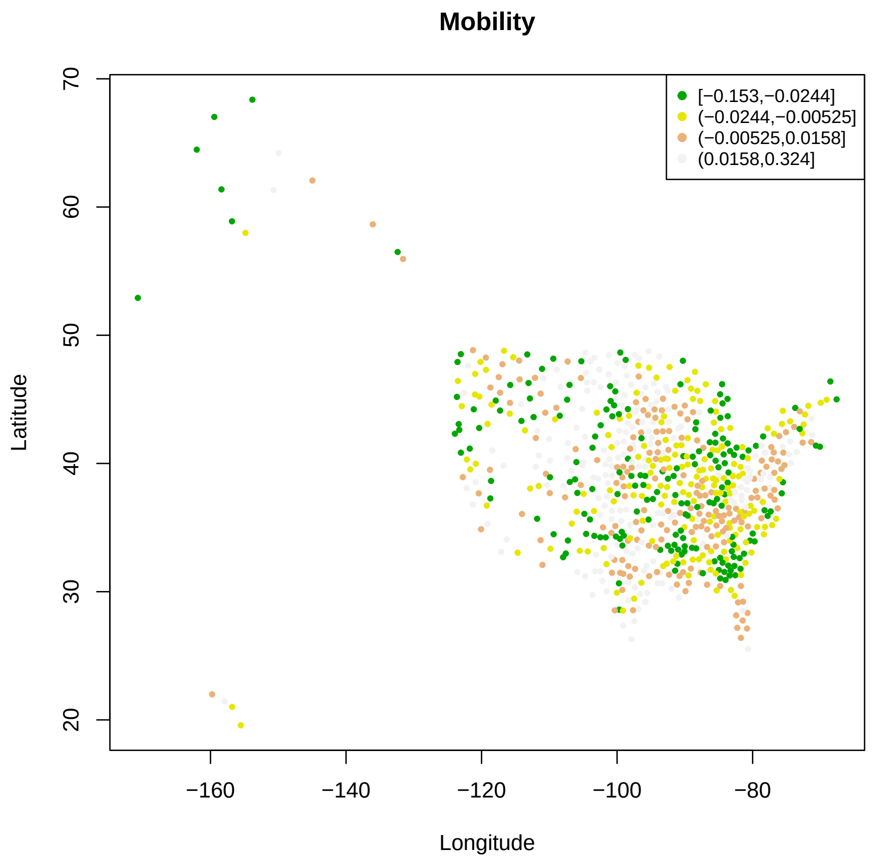


FIGURE 14.3: *Map of residuals from a basline linear regression of the rate of economic mobility on the fraction of workers with short commutes.*



```
residuals.state.map <- mapper(residuals(mob.state), levels = 4, pch = 19, cex = 0.5,
    xlab = "Longitude", ylab = "Latitude", main = "Mobility")
legend("topright", legend = levels(residuals.state.map$categories), pch = 19, col = terrain.colors(
    cex = 0.8)
```

FIGURE 14.4: *Map of the residuals for the model with state-level contrasts. (See Figure 14.2 for the* `mapper` *function.)*



```
residuals.dixie.map <- mapper(residuals(mob.dixie), levels = 4, pch = 19, cex = 0.5,
    xlab = "Longitude", ylab = "Latitude", main = "Mobility")
legend("topright", legend = levels(residuals.dixie.map$categories), pch = 19, col = terrain.colors(4),
    cex = 0.8)
```

FIGURE 14.5: *Map of the residuals from the model based on* `Commute`*, and a categorical contrast between the old Confederacy and the rest of the country.*

14.4 Further Reading

Polynomial regression and categorical predictors are both ancient topics; I don't know who first introduced either.

The above discussion has assumed that when we use a polynomial, we use the *same* polynomial for all values of X_i. An alternative is to use different, low-order polynomials in different regions. If these piecewise polynomial functions are required to be continuous, they are called **splines**, and regression with splines will occupy us for much of 402, because it gives us ways to tackle lots of the issues with polynomials, like over-fitting (Shalizi, forthcoming, chs. 8 and 9). Personally, I have found splines to almost always be a better tool than polynomial regression, but they do demand a bit more math.

The matter of "adjusted effects" and causal inference will occupy us for about the last quarter of 36-402.

Tutz (2012) is a thorough and modern survey of regression with categorical *response* variables. We will go over this in some detail in 402, but his book covers many topics we won't have time for.

Winship and Mare (1984) proposes some interesting techniques for dealing with ordinal variables, under the (strong) assumption that they arise from taking continuous variables and breaking them into discrete categories. This seems to require rather strong assumptions about the measurement process. Another direction we could go would be to estimate a separate contrast for each level of an ordinal variable (except the lowest), but require these to be either all increasing or all decreasing, so the response to the ordinal variable was monotone. This would mean solving a *constrained* least squares (or maximum likelihood) problem to get the estimates, not an unconstrained on. Worse, the constraints would be a somewhat awkward set of inequalities. Still, it's do-able in principle, though I don't know of a straightforward R implementation.

Analysis of variance models were introduced by R. A. Fisher, probably the greatest statistician who ever lived, in connection with problems in genetics and in designing and interpreting experiments. They have given rise to a huge literature and an elaborate system of notation and terminology, much of which boils down to short-cuts for computing regression estimates when the design matrix $\mathbf{x}$ has very special structure. As I said, there were many decades when such short-cuts were vital, but I am frankly skeptical how much value these techniques retain in the present day. In the interest of balance, see Gelman (2005) for a contrary view.

I mentioned that one reason to use polynomials is that any well-behaved function can be approximated arbitrarily closely by polynomials of sufficiently high degree. Obviously "well-behaved" needs a proper definition, as (perhaps less obviously) does "approximated arbitrarily closely". What I had in mind was the Stone-Weierstrass theorem, which states that you can pick any continuous function f, interval $[a, b]$, and tolerance $\epsilon > 0$ you like, and I can find some polynomial which is within ϵ of f everywhere on the interval,

$$\max_{a \leq x \leq b} \left| f(x) - \sum_{j=1}^{d} \gamma_j x^j \right| \leq \epsilon$$

provided there is no limit on the order d or the magnitude of the coefficients γ_j. This is a standard result of real analysis, which will be found in almost textbook on that subject, or on functional analysis or approximation theory. There are parallel results for other function bases.

14.5 Exercises

1. Consider regressing Y on a binary categorical variable B, plus some other predictors. Suppose we switch which level is the reference category and which one is contrasted with it. Show that this produces the following changes to the parameters, and leaves all the others unchanged:

$$\beta_0 \rightarrow \beta_0 + \beta_B \quad (14.1)$$

$$\beta_B \rightarrow -\beta_B \quad (14.2)$$

   Hint: Show that the change to the indicator variable is $X_B \rightarrow 1 - X_B$.

2. Consider again regressing Y on a binary variable B, plus some other predictors, and estimating all coefficients by least squares. Show that the average of all residuals where $X_B = 1$ must be exactly 0, as must the average of all residuals where $X_B = 0$. *Hint:* Use the estimating equations to show $\sum_i e_i = 0$, $\sum_i e_i x_{Bi} = 0$, and algebra to show $\sum_i e_i(1 - x_{Bi}) = 0$.

Chapter 15

Multicollinearity

15.1 Why Collinearity Is a Problem

Remember our formula for the estimated coefficients in a multiple linear regression:

$$\widehat{\beta} = (\mathbf{x}^T \mathbf{x})^{-1} \mathbf{x}^T \mathbf{y}$$

This is obviously going to lead to problems if $\mathbf{x}^T \mathbf{x}$ isn't invertible. Similarly, the variance of the estimates,

$$\mathrm{Var}\left[\widehat{\beta}\right] = \sigma^2 (\mathbf{x}^T \mathbf{x})^{-1}$$

will blow up when $\mathbf{x}^T \mathbf{x}$ is singular. If that matrix isn't exactly singular, but is close to being non-invertible, the variances will become huge.

There are several equivalent conditions for any square matrix, say $\mathbf{u}$, to be singular or non-invertible:

- The determinant $\det \mathbf{u}$ or $|\mathbf{u}|$ is 0.
- At least one eigenvalue[1] of u is 0. (This is because the determinant of a matrix is the product of its eigenvalues.)
- $\mathbf{u}$ is **rank deficient**, meaning that one or more of its columns (or rows) is equal to a linear combination of the other rows[2].

Since we're not concerned with any old square matrix, but specifically with $\mathbf{x}^T \mathbf{x}$, we have an additional equivalent condition:

- $\mathbf{x}$ is **column-rank** deficient, meaning one or more of its columns is equal to a linear combination of the others.

[1] You learned about eigenvalues and eigenvectors in linear algebra; if you are rusty, now is an excellent time to refresh your memory.

[2] The equivalence of this condition to the others is not at all obvious, but, again, is proved in linear algebra.

The last explains why we call this problem **collinearity**: it looks like we have p different predictor variables, but really some of them are linear combinations of the others, so they don't add any information. The real number of distinct variables is $q < p$, the column rank of $\mathbf{x}$. If the exact linear relationship holds among more than two variables, we talk about **multicollinearity**; **collinearity** can refer either to the general situation of a linear dependence among the predictors, or, by contrast to multicollinearity, a linear relationship among just two of the predictors.

Again, if there isn't an *exact* linear relationship among the predictors, but they're close to one, $\mathbf{x}^T\mathbf{x}$ will be invertible, but $(\mathbf{x}^T\mathbf{x})^{-1}$ will be huge, and the variances of the estimated coefficients will be enormous. This can make it very hard to say anything at all precise about the coefficients, but that's not *necessarily* a problem.

15.1.1 Dealing with Collinearity by Deleting Variables

Since not all of the p variables are actually contributing information, a natural way of dealing with collinearity is to drop some variables from the model. If you want to do this, you should think very carefully about *which* variable to delete. As a concrete example: if we try to include all of a student's grades as predictors, as well as their over-all GPA, we'll have a problem with collinearity (since GPA is a linear function of the grades). But depending on what we want to predict, it might make more sense to use just the GPA, dropping all the individual grades, or to include the individual grades and drop the average[3].

15.1.2 Diagnosing Collinearity Among Pairs of Variables

Linear relationships between pairs of variables are fairly easy to diagnose: we make the pairs plot of all the variables, and we see if any of them fall on a straight line, or close to one. Unless the number of variables is huge, this is by far the best method. If the number of variables *is* huge, look at the correlation matrix, and worry about any entry off the diagonal which is (nearly) ± 1.

15.1.3 Why Multicollinearity Is Harder

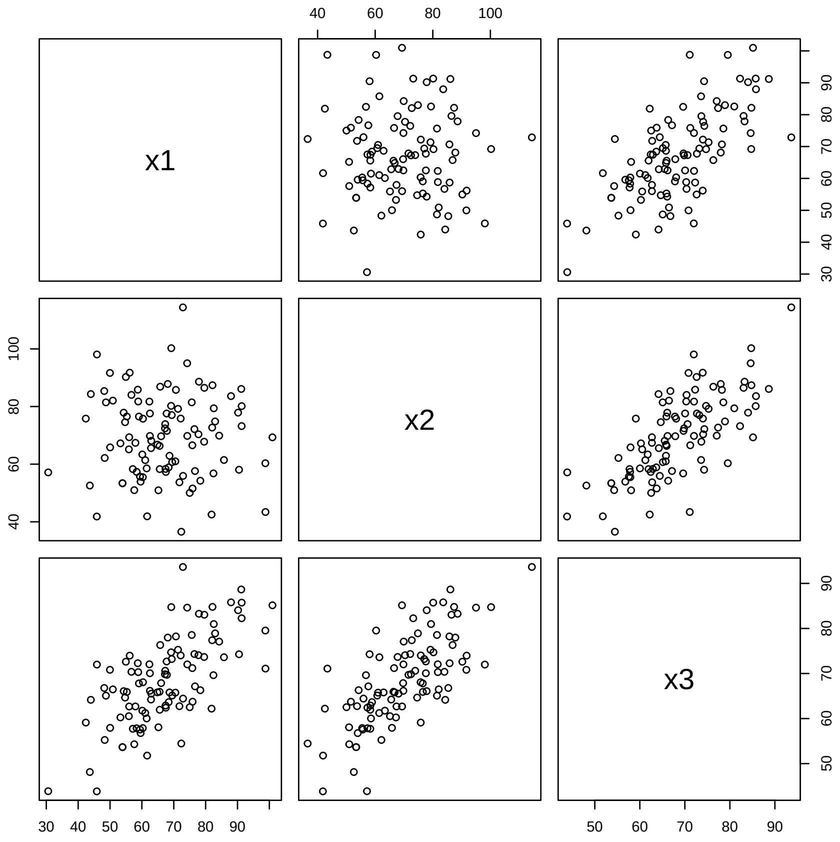
A multicollinear relationship involving three or more variables might be totally invisible on a pairs plot. For instance, suppose X_1 and X_2 are independent Gaussians, of equal variance σ^2, and X_3 is their average, $X_3 = (X_1 + X_2)/2$. The correlation between

[3]One could also drop just one of the individual class grades from the average, but it's harder to think of a scenario where that makes sense.

X_1 and X_3 is

$$\begin{aligned}
\mathrm{Cor}(X_1, X_3) &= \frac{\mathrm{Cov}[X_1, X_3]}{\sqrt{\mathrm{Var}[X_1]\mathrm{Var}[X_3]}} && (15.1)\\
&= \frac{\mathrm{Cov}[X_1, (X_1 + X_2)/2]}{\sqrt{\sigma^2\sigma^2/2}} && (15.2)\\
&= \frac{\sigma^2/2}{\sigma^2/\sqrt{2}} && (15.3)\\
&= \frac{1}{\sqrt{2}} && (15.4)
\end{aligned}$$

This is also the correlation between X_2 and X_3. A correlation of $1/\sqrt{2}$ isn't trivial, but is hardly perfect, and doesn't really distinguish itself on a pairs plot (Figure 15.1).



```
##              x1          x2        x3
## x1  1.00000000 -0.02227466 0.6783007
## x2 -0.02227466  1.00000000 0.7194932
## x3  0.67830074  0.71949317 1.0000000
```

```
# Simulation: two independent Gaussians
x1 <- rnorm(100, mean = 70, sd = 15)
x2 <- rnorm(100, mean = 70, sd = 15)
# Add in a linear combination of X1 and X2
x3 <- (x1 + x2)/2
pairs(cbind(x1, x2, x3))
cor(cbind(x1, x2, x3))
```

FIGURE 15.1: *Illustration of a perfect* multi-*collinear relationship might not show up on a pairs plot or in a correlation matrix.*

15.1.4 Geometric Perspective

The predictors $X_1, \ldots X_p$ form a p-dimensional random vector $\mathbf{X}$. Ordinarily, we expect this random vector to be scattered throughout p-dimensional space. When we have collinearity (or multicollinearity), the vectors are actually confined to a lower-dimensional subspace. The **column rank** of a matrix is the number of linearly independent columns it has. If $\mathbf{x}$ has column rank $q < p$, then the data vectors are confined to a q-dimensional subspace. It *looks* like we've got p different variables, but really by a change of coordinates we could get away with just q of them.

15.2 Variance Inflation Factors

If the predictors are correlated with each other, the standard errors of the coefficient estimates will be bigger than if the predictors were uncorrelated.

If the predictors were uncorrelated, the variance of $\hat{\beta}_i$ would be

$$\mathrm{Var}\left[\hat{\beta}_i\right] = \frac{\sigma^2}{n s^2_{X_i}} \tag{15.5}$$

just as it is in a simple linear regression. With correlated predictors, however, we have to use our general formula for the least squares:

$$\mathrm{Var}\left[\hat{\beta}_i\right] = \sigma^2 (\mathbf{x}^T\mathbf{x})^{-1}_{i+1,i+1} \tag{15.6}$$

(Why are the subscripts on the matrix $i+1$ instead of i?) The ratio between Eqs. 15.6 and 15.5 is the **variance inflation factor** for the i^{th} coefficient, VIF_i. The average of the variance inflation factors across all predictors is often written $\overline{VIF}$, or just VIF.

Folklore says that $VIF_i > 10$ indicates "serious" multicollinearity for the predictor. I have been unable to discover who first proposed this threshold, or what the justification for it is. It is also quite unclear what to do about this. Large variance inflation factors do not, after all, violate any model assumptions.

15.2.1 Why $VIF_i \geq 1$

Let's take the case where $p = 2$, so $\mathbf{x}^T\mathbf{x}$ is a 3×3 matrix. As you saw in the homework,

$$\frac{1}{n}\mathbf{x}^T\mathbf{x} = \left[\begin{array}{ccc} 1 & \overline{x_1} & \overline{x_2} \\ \overline{x_1} & \overline{x_1^2} & \overline{x_1 x_2} \\ \overline{x_2} & \overline{x_1 x_2} & \overline{x_2^2} \end{array}\right] \tag{15.7}$$

After tedious but straightforward algebra[4], we get for the inverse (deep breath)

$$\begin{aligned}
&\left(\frac{1}{n}\mathbf{x}^T\mathbf{x}\right)^{-1} \\
&= \frac{1}{\widehat{\mathrm{Var}}[X_1]\widehat{\mathrm{Var}}[X_2]-\widehat{\mathrm{Cov}}[X_1,X_2]^2} \\
&\times \left[\begin{array}{ccc}
\widehat{\mathrm{Var}}[X_1]\widehat{\mathrm{Var}}[X_2]-\widehat{\mathrm{Cov}}[X_1,X_2]^2+\widehat{\mathrm{Var}}[\overline{x_2}X_1-\overline{x_1}X_2] & \overline{x_1}\widehat{\mathrm{Var}}[X_2]-\widehat{\mathrm{Cov}}[X_1,X_2]\overline{x_2} & \overline{x_1}\widehat{\mathrm{Cov}}[X_1,X_2]-\widehat{\mathrm{Var}}[X_1] \\
\overline{x_1}\widehat{\mathrm{Var}}[X_2]-\widehat{\mathrm{Cov}}[X_1,X_2]\overline{x_2} & \widehat{\mathrm{Var}}[X_2] & -\widehat{\mathrm{Cov}}[X_1,X_2] \\
\overline{x_1}\widehat{\mathrm{Cov}}[X_1,X_2]-\widehat{\mathrm{Var}}[X_1]\overline{x_2} & -\widehat{\mathrm{Cov}}[X_1,X_2] & \widehat{\mathrm{Var}}[X_1]
\end{array} \right.
\end{aligned}$$

where the hats on the variances and covariances indicate that they are sample, not population, quantities.

Notice that the pre-factor to the matrix, which is the determinant of $n^{-1}\mathbf{x}^T\mathbf{x}$, blows up when X_1 and X_2 are either perfectly correlated or perfectly anti-correlated — which is as it should be, since then we'll have exact collinearity.

The variances of the estimated slopes are, using this inverse,

$$\mathrm{Var}\left[\hat{\beta}_1\right] = \frac{\sigma^2}{n}\frac{\widehat{\mathrm{Var}}[X_2]}{\widehat{\mathrm{Var}}[X_1]\widehat{\mathrm{Var}}[X_2]-\widehat{\mathrm{Cov}}[X_1,X_2]^2} = \frac{\sigma^2}{n(\widehat{\mathrm{Var}}[X_1]-\widehat{\mathrm{Cov}}[X_1,X_2]^2/\widehat{\mathrm{Var}}[X_2])}$$

and

$$\mathrm{Var}\left[\hat{\beta}_2\right] = \frac{\sigma^2}{n}\frac{\widehat{\mathrm{Var}}[X_1]}{\widehat{\mathrm{Var}}[X_1]\widehat{\mathrm{Var}}[X_2]-\widehat{\mathrm{Cov}}[X_1,X_2]^2} = \frac{\sigma^2}{n(\widehat{\mathrm{Var}}[X_2]-\widehat{\mathrm{Cov}}[X_1,X_2]^2/\widehat{\mathrm{Var}}[X_1])}$$

Notice that if $\widehat{\mathrm{Cov}}[X_1,X_2] = 0$, these reduce to

$$\mathrm{Var}\left[\hat{\beta}_1\right] = \frac{\sigma^2}{n\widehat{\mathrm{Var}}[X_1]},\ \mathrm{Var}\left[\hat{\beta}_2\right] = \frac{\sigma^2}{n\widehat{\mathrm{Var}}[X_2]}$$

exactly as we'd see in simple linear regressions. When covariance is present, however, regardless of its sign, it increases the variance of the estimates.

With a great deal of even more tedious algebra, it can be shown that this isn't just a weird fact about the $p = 2$ case, but is true generically. The variance inflation factor for X_i can be found by regressing X_i on all of the other X_j, computing the R^2 of this regression[5], say R^2_i, and setting $VIF_i = 1/(1-R^2_i)$.[6] The consequence is that $VIF_i \geq 1$, with the variance inflation factor increasing as X_i becomes more correlated with some linear combination of the other predictors.

[4]At least, if you remember how to calculate the determinant of a matrix, a matter on which I evidently had a brain-fault this afternoon.

[5]I'd admit this was an exception to my claim that R^2 is at best useless, except that we can get the exact same number, without running all these regressions, just by inverting $\mathbf{x}^T\mathbf{x}$.

[6]The trick to showing this involves relating the co-factors which appear when we're inverting $n^{-1}\mathbf{x}^T\mathbf{x}$ to the coefficients in the regression of X_i on all the other X_j, followed by a mess of book-keeping.

15.3 Matrix-Geometric Perspective on Multicollinearity

Multicollinearity means that there exists (at least) one set of constants $a_0, a_1, \ldots a_p$, $a_1, \ldots a_p$ not all zero, such that

$$a_1X_1 + a_2X_2 + \ldots a_pX_p = \sum_{i=1}^{p} a_iX_i = a_0$$

To simplify this, let's introduce the $p \times 1$ matrix $\mathbf{a} = \left[\begin{array}{c} a_1 \\ \vdots \\ a_p \end{array}\right]$, so we can write multicollinearity as

$$\mathbf{a}^T\mathbf{X} = a_0$$

for $\mathbf{a} \neq 0$.

If this equation holds, then

$$\mathrm{Var}\left[\mathbf{a}^T\mathbf{X}\right] = \mathrm{Var}\left[\sum_{i=1}^{p} a_iX_i\right] = \mathrm{Var}\left[a_0\right] = 0$$

Conversely, if $\mathrm{Var}\left[\mathbf{a}^T\mathbf{X}\right] = 0$, then $\mathbf{a}^T\mathbf{X}$ must be equal to some constant, which we can call a_0. So multicollinearity is equivalent to the existence of a vector $\mathbf{a} \neq 0$ where

$$\mathrm{Var}\left[\mathbf{a}^T\mathbf{X}\right] = 0$$

I make these observations because we are old hands now at the variances of weighted sums.

$$\begin{aligned} \mathrm{Var}\left[\mathbf{a}^T\mathbf{X}\right] &= \mathrm{Var}\left[\sum_{i=1}^{p} a_iX_i\right] & (15.10) \\ &= \sum_{i=1}^{p}\sum_{j=1}^{p} a_ia_j\mathrm{Cov}\left[X_i, X_j\right] & (15.11) \\ &= \mathbf{a}^T\mathrm{Var}\left[\mathbf{X}\right]\mathbf{a} & (15.12) \end{aligned}$$

Multicollinearity therefore means the equation

$$\mathbf{a}^T\mathrm{Var}\left[\mathbf{X}\right]\mathbf{a} = 0$$

has a solution $\mathbf{a} \neq 0$.

Solving a quadratic equation in matrices probably does not sound like much fun, but this is where we appeal to results in linear algebra[7]. $\mathrm{Var}\left[\mathbf{X}\right]$ is a very special matrix: it is square ($p \times p$), symmetric, and positive-definite, meaning that $\mathbf{a}^T\mathrm{Var}\left[\mathbf{X}\right]\mathbf{a} \geq 0$. (Since, after all, that expression is the variance of the scalar $\sum_{i=1}^{p} a_iX_i$, and variances of scalars are ≥ 0.) We may therefore appeal to the *spectral* or *eigendecomposition* theorem of linear algebra to assert the following:

[7]This is also a big part of why we make you take linear algebra.

1. There are p different $p \times 1$ vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots \mathbf{v}_p$, the **eigenvectors** of $\mathrm{Var}[\mathbf{X}]$, such that

$$\mathrm{Var}[\mathbf{X}]\mathbf{v}_i = \lambda_i \mathbf{v}_i$$

   for scalar constants $\lambda_1, \lambda_2, \ldots \lambda_p$, the **eigenvalues** of $\mathrm{Var}[\mathbf{X}]$. The ordering of the eigenvalues and eigenvectors is arbitrary, but it is conventional to arrange them so that $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_p$.

2. The eigenvalues are all ≥ 0. (Some of them may be equal to each other; these are called **repeated**, **multiple** or **degenerate** eigenvalues.)

3. The eigenvectors can be chosen so that they all have length 1, and are orthogonal to each other, so $\mathbf{v}_j^T \mathbf{v}_i = \delta_{ij}$.

4. Any vector can be re-written as a sum of eigenvectors:

$$\mathbf{a} = \sum_{i=1}^{p} (\mathbf{a}^T \mathbf{v}_i)\mathbf{v}_i$$

   (Here I have used the parentheses to eliminate any ambiguity about the order in which the matrices are to be multiplied; $\mathbf{a}^T \mathbf{v}_i$ is always a scalar.)

5. $\mathrm{Var}[\mathbf{X}]$ can be expressed as

$$\mathrm{Var}[\mathbf{X}] = \mathbf{V}\mathbf{D}\mathbf{V}^T$$

   where $\mathbf{V}$ is the matrix whose i^{th} column is $\mathbf{v}_i$, (and so $\mathbf{V}^T$ is the matrix where $\mathbf{v}_i$ is the i^{th} column), and $\mathbf{D}$ is the diagonal matrix whose entries are $\lambda_1, \lambda_2, \ldots \lambda_p$.

Suppose that one or more of the eigenvalues are zero. Since we've put them in order, this means that the positive eigenvalues are $\lambda_1, \ldots \lambda_q$ (for some $q < p$), and $\lambda_{q+1}, \ldots \lambda_p$ are all zero. It follows that $\mathbf{v}_{q+1}, \ldots \mathbf{v}_p$ all give us linear combinations of the X_i which are multicollinear. So a sufficient condition for multicollinearity is that $\mathrm{Var}[\mathbf{X}]$ have zero eigenvalues.

Conversely, suppose $\mathbf{a}^T \mathrm{Var}[\mathbf{X}]\mathbf{a} = 0$, and $\mathbf{a} \neq 0$. Let's re-express this using the eigendecomposition.

$$\begin{aligned}
\mathrm{Var}[\mathbf{X}]\mathbf{a} &= \mathrm{Var}[\mathbf{X}]\sum_{i=1}^{p}(\mathbf{a}^T\mathbf{v}_i)\mathbf{v}_i && (15.13)\\
&= \sum_{i=1}^{p}(\mathbf{a}^T\mathbf{v}_i)\mathrm{Var}[\mathbf{X}]\mathbf{v}_i && (15.14)\\
&= \sum_{i=1}^{p}(\mathbf{a}^T\mathbf{v}_i)\lambda_i\mathbf{v}_i && (15.15)\\
\mathbf{a}^T\mathrm{Var}[\mathbf{X}]\mathbf{a} &= \left(\sum_{j=1}^{p}(\mathbf{a}^T\mathbf{v}_j)\mathbf{v}_j\right)^T\sum_{i=1}^{p}(\mathbf{a}^T\mathbf{v}_i)\lambda_i\mathbf{v}_i && (15.16)\\
&= \sum_{i=1}^{p}\sum_{j=1}^{p}(\mathbf{a}^T\mathbf{v}_i)(\mathbf{a}^T\mathbf{v}_j)\mathbf{v}_j^T\mathbf{v}_i && (15.17)\\
&= \sum_{i=1}^{p}(\mathbf{a}^T\mathbf{v}_i)^2\lambda_i && (15.18)
\end{aligned}$$

Since $(\mathbf{a}^T\mathbf{v}_i)^2 \geq 0$, the only way the whole sum can be zero is if $(\mathbf{a}^T\mathbf{v}_i)^2 > 0$ only when $\lambda_i = 0$.

We have therefore established the following:

1. The predictors are multi-collinear if and only if $\mathrm{Var}[\mathbf{X}]$ has zero eigenvalues.
2. Every multi-collinear combination of the predictors is either an eigenvector of $\mathrm{Var}[\mathbf{X}]$ with zero eigenvalue, or a linear combination of such eigenvectors.

15.3.1 The Geometric View

Every eigenvector of $\mathrm{Var}[\mathbf{X}]$ points out a direction in the space of predictors. The leading eigenvector $\mathbf{v}_1$, the one going along with the largest eigenvalue, points out the direction of highest variance (and that variance is λ_1). The next-to-leading eigenvector, $\mathbf{v}_2$, points out the direction orthogonal to $\mathbf{v}_1$ which has the highest variance, and so forth down the line. The eigenvectors of $\mathrm{Var}[\mathbf{X}]$ are also called the **principal components** of the predictors, because of their role as the directions of maximum variance.

The eigenvectors going along with zero eigenvalues point out directions in the predictor space along which there is no variance, precisely because those directions amount to weighted sums of the original variables which equal constants. The q non-zero eigenvalues mark out the q-dimensional subspace in which all the data vectors lie. If $q < p$, then the predictors are rank-deficient, and the rank of $\mathbf{x}$ is just q.

15.3.2 Finding the Eigendecomposition

Because finding eigenvalues and eigenvectors of matrices is so useful for so many situations, mathematicians and computer scientists have devoted incredible efforts over

```
# Simulation: two independent Gaussians
x1 <- rnorm(100, mean = 70, sd = 15)
x2 <- rnorm(100, mean = 70, sd = 15)
# Add in a linear combination of X1 and X2
x3 <- (x1 + x2)/2
# X4 is somewhat correlated with X1 but not relevant to Y
x4 <- x1 + runif(100, min = -100, max = 100)
# Y is a linear combination of the X's plus noise
y <- 0.7 * x1 + 0.3 * x2 + rnorm(100, mean = 0, sd = sqrt(15))
df <- data.frame(x1 = x1, x2 = x2, x3 = x3, x4 = x4, y = y)
```

FIGURE 15.2: *Small simulation illustrating exact collinearity.*

the last two hundred years to fact, precise algorithms for computing them. This is not the place to go over how those algorithms work; it is the place to say that much of the fruit of those centuries of effort is embodied in the linear algebra packages R uses. Thus, when you call

```
eigen(A)
```

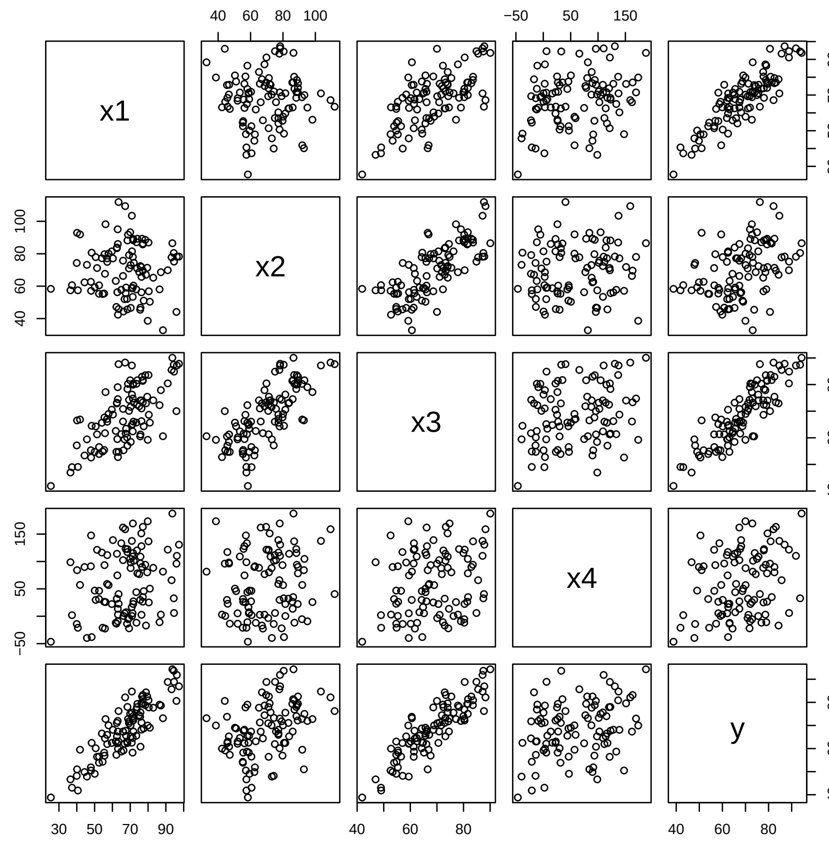
you get back a list, containing the eigenvalues of the matrix `A` (in a vector), and its eigenvectors (in a matrix), and this is both a very fast and a very reliable calculation. If your matrix has very special structure (e.g., it's sparse, meaning almost all its entries are zero), there are more specialized packages adapted to your needs, but we don't pursue this further here; for most data-analytic purposes, ordinary `eigen` will do.

15.3.3 Using the Eigendecomposition

1. Find the eigenvalues and eigenvectors.
2. If any eigenvalues are zero, the data is multicollinear; if any are very close to zero, the data is nearly multicollinear.
3. Examine the corresponding eigenvectors. These indicate the linear combinations of variables which equal constants (or are nearly constant if the eigenvalue is only nearly zero). Ideally, these will be combinations of a reasonably small number of variables (i.e., most of the entries in the eigenvector will be zero), so you can ask whether there are substantive reasons to delete one or more of those predictors.

15.3.3.1 Example

I'll make up some data which displays exact multi-collinearity. Let's say that X_1 and X_2 are both Gaussian with mean 70 and standard deviation 15, and are uncorrelated; that $X_3 = (X_1 + X_2)/2$; and that $Y = 0.7X_1 + 0.3X_2 + \epsilon$, with $\epsilon \sim N(0, 15)$.



```
##              x1           x2        x3        x4         y
## x1  1.000000000 -0.008843601 0.6641534 0.2577387 0.8717515
## x2 -0.008843601  1.000000000 0.7416936 0.1177607 0.3946726
## x3  0.664153379  0.741693598 1.0000000 0.2609231 0.8798078
## x4  0.257738729  0.117760741 0.2609231 1.0000000 0.3036906
## y   0.871751501  0.394672569 0.8798078 0.3036906 1.0000000
```

```
pairs(df)
cor(df)
```

FIGURE 15.3: *Pairs plot and correlation matrix for the example of Figure 15.2. Notice that neither the pairs plot nor the correlation matrix reveals a problem, which is because it only arises when considering X_1, X_2, X_3 at once.*

```
# Create the variance matrix of the predictor variables
var.x <- var(df[, c("x1", "x2", "x3", "x4")])
# Find the eigenvalues and eigenvectors
var.x.eigen <- eigen(var.x)
# Which eigenvalues are (nearly) 0?
(zero.eigenvals <- which(var.x.eigen$values < 1e-12))
## [1] 4
# Display the corresponding vectors
(zero.eigenvectors <- var.x.eigen$vectors[, zero.eigenvals])
## [1]  4.082483e-01  4.082483e-01 -8.164966e-01  2.220446e-16
```

FIGURE 15.4: *Example of using the eigenvectors of* $\mathrm{Var}[X]$ *to find collinear combinations of the predictor variables. Here, what this suggests is that* $-X_1 - X_2 + 2X_3 = \text{constant}$. *This is correct, since* $X_3 = (X_1 + X_2)/2$*, but the eigen-decomposition didn't* know *this; it discovered it.*

15.3.4 Principal Components Regression

Let's define some new variables:

$$\begin{aligned} W_1 &= \mathbf{v}_1^T X & (15.19) \\ W_i &= \mathbf{v}_i^T X & (15.20) \\ W_p &= \mathbf{v}_p^T X & (15.21) \\ & & (15.22) \end{aligned}$$

W_1 is the projection of the original data vector X onto the leading eigenvector, or the **principal component**. It is called the **score** on the first principal component. It has a higher (sample) variance than any other linear function of the original predictors. W_2 is the projection or score on the second principle component. It has more variance than any other linear combination of the original predictors which is uncorrelated with W_1. In fact, $\widehat{\text{Cov}}\left[W_i, W_j\right] = 0$ if $i \neq j$.

In **principle components regression**, we pick some $k \leq p$ and use the model

$$Y = \gamma_0 + \gamma_1 W_1 + \ldots \gamma_k W_k + \epsilon$$

where as usual we presume ϵ has expectation 0, constant variance, and no correlation from one observation to another. (I use the Greek letter γ, instead of β, to emphasize that these coefficients are *not* directly comparable to those of our original linear model.) We are regressing not on our original variables, but on uncorrelated linear combinations of those variables.

If $k = p$, then we get exactly the same predictions and fitted values as in the original linear model, though the coefficients aren't the same. This would amount to doing a change of coordinates in the space of predictors, so that all of the new coordinates were uncorrelated, but wouldn't otherwise change anything.

If there are only $q < p$ non-zero eigenvalues, we should not use $k > q$. Using $k = q$ uses all the linear combinations of the original predictors which aren't collinear. However, we might deliberately pick $k < q$ so as to simplify our model. As I said above, the principal components are the directions in the predictor space with the highest variance, so by using a small k we confine ourselves to those directions, and ignore all the other aspects of our original predictors. This may introduce bias, but should reduce the variance in our predictions. (Remember that the variance in our coefficient estimates, and so in our predictions, goes down with the variance of the predictor variables.)

There are a number of things to be said about principal components regression.

1. We need some way to pick k. The in-sample MSE will decrease as k grows (why?), but this might not be a good guide to out-of-sample predictions, or to whether the modeling assumptions are fulfilled.

2. The PC regression can be hard to interpret.

The last point needs some elaboration. Each one of the principal components is a linear combination of the original variables. Sometimes these are easy to interpret, other times their meaning (if they have any) is thoroughly obscure. Whether this matters depends very much on how deeply committed you are to interpreting the coefficients.

As for picking k, there are two (potentially) rival objectives. One is to pick the number of components which will let us predict well. The in-sample mean squared error *has* to decrease as k grows, so we would really like some measure of actual out-of-sample or generalization error; the cross-validation method I will describe below is applicable, but there are other potentially-applicable techniques. The other objective is to have a set of variables which satisfy the assumptions of the multiple linear regression model. In my experience, it is not very common for principal components regression to actually satisfy the modeling assumptions, but it can work surprisingly well as a predictive tool anyway.

15.4 Ridge Regression

The real problem with collinearity is that when it happens, there isn't a *unique* solution to the estimating equations. There are rather infinitely many solutions, which all give the minimum mean squared error. It feels perverse, somehow, to get rid of predictors because they give us too many models which fit too well. A better response is to pick *one* of these solutions, by adding some other criterion we'd like to see in the model.

There are many ways to do this, but one which works well in practice is the following: all else being equal, we prefer models with smaller slopes, ones closer to zero. Specifically, let's say that we prefer the length of the coefficient vector, $||\beta||$, to be small. Now, at least abstractly, we have a situation like that shown in Figure 15.5. The black line marks out all of the β_1, β_2 combinations which give us exactly the same mean squared error. They all give the same error because of a collinearity between X_1 and X_2. But there is a single point on the black line which comes closest to the origin — it touches the solid grey circle. Other points on the line, while they have equal MSEs, have larger $||\beta||$ (they lie on one of the dashed grey circles), so we don't use them.

What if everything else isn't equal? (Say, for instance, that the data are only *nearly* collinear.) We'll need some way to trade off having a smaller MSE against having a smaller vector of coefficients. Since we're looking at *squared* error, I hope it is somewhat plausible that we should also look at the *squared* length of the coefficient vector; if you don't buy that, you can at least take my word for it that it simplifies the math.

Specifically, let's replace our old optimization problem

$$\min_{\beta} \frac{1}{n}(\mathbf{y}-\mathbf{x}\mathbf{b})^T(\mathbf{y}-\mathbf{x}\mathbf{b})$$

with a new, *penalized* optimization problem

$$\min_{\beta} \frac{1}{n}(\mathbf{y}-\mathbf{x}\mathbf{b})^T(\mathbf{y}-\mathbf{x}\mathbf{b}) - \frac{\lambda}{n}||\mathbf{b}||^2$$

```
# Add a circle to an existing plot R, bizarrely, does not have any built-in
# function for this Inputs: x coordinate of center; y coordinate of center;
# radius; number of equiangular steps; additional graphical parameters Outputs:
# none Side-effects: a circle is added to the existing plot
circle <- function(x0, y0, r, n = 1000, ...) {
    theta <- seq(from = 0, to = 2 * pi, length.out = n)  # Angles
    x <- x0 + r * cos(theta)  # x coordinates
    y <- y0 + r * sin(theta)  # y coordinates
    lines(x, y, ...)  # Draw the lines connecting all the points, in order
}
plot(0, type = "n", xlab = expression(beta[1]), ylab = expression(beta[2]), xlim = c(-10,
    10), ylim = c(-10, 10))
abline(a = 10, b = -2)
points(0, 0)
circle(0, 0, sqrt(20), col = "grey")
points(4, 2, col = "black", pch = 19)
circle(0, 0, 5, col = "grey", lty = "dashed")
circle(0, 0, 6, col = "grey", lty = "dashed")
```

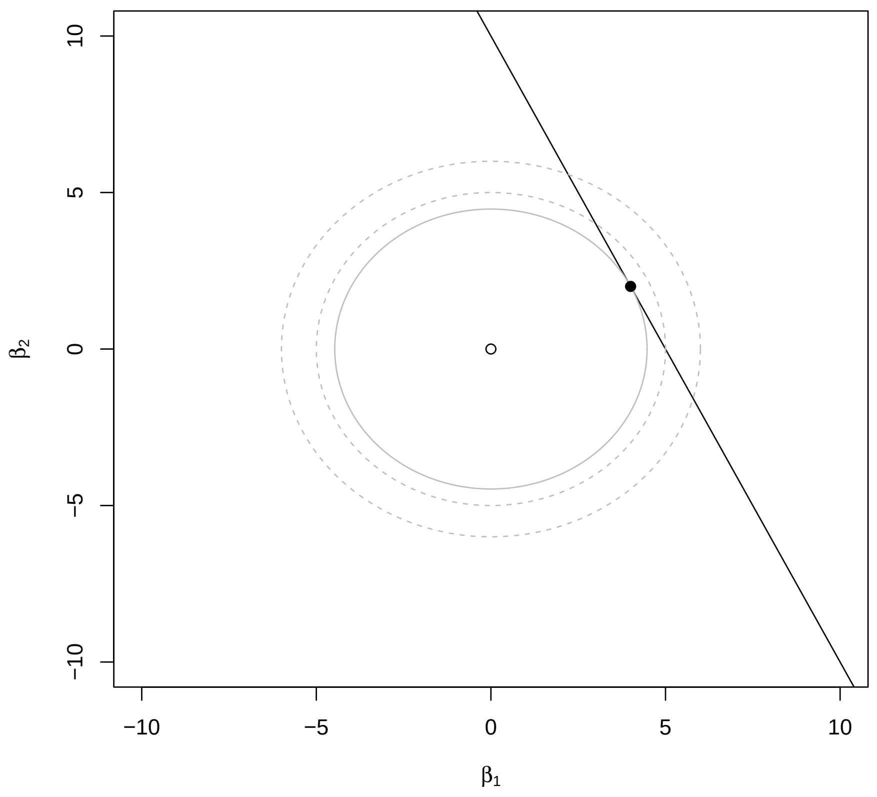


FIGURE 15.5: *Geometry of ridge regression when the predictors are collinear. The black line shows all the combinations of β_1 and β_2 which minimize the MSE. We chose the coefficient vector (the black point) which comes closest to the origin (the dot). Equivalently, this is the parameter vector with the smallest MSE among all the points at equal distance from the origin (solid grey circle). Other coefficient vectors either have a worse MSE (they don't lie on the black line), or are further from the origin (they lie on one of the dashed grey circles).*

Here the **penalty factor** $\lambda > 0$ tells us the rate at which we're willing to make a trade-off between having a small mean squared error and having a short vector of coefficients[8]. We'd accept $||\mathbf{b}||^2$ growing by 1 if it reduced the MSE by more than λ/n. We'll come back later to how to pick λ.

It's easy to pose this optimization problem: can we actually solve it, and is the solution good for anything? Solving is actually straightforward. We can re-write $||\mathbf{b}||^2$ as, in matrix notation, $\mathbf{b}^T\mathbf{b}$, so the gradient is

$$\nabla_{\mathbf{b}}\left(\frac{1}{n}(\mathbf{y}-\mathbf{x}\mathbf{b})^T(\mathbf{y}-\mathbf{x}\mathbf{b})+\frac{\lambda}{n}\mathbf{b}^T\mathbf{b}\right)=\frac{2}{n}\left(-\mathbf{x}^T\mathbf{y}+\mathbf{x}^T\mathbf{x}\mathbf{b}+\lambda\mathbf{b}\right)$$

Set this to zero at the optimum, $\tilde{\beta}_\lambda$,

$$\mathbf{x}^T\mathbf{y}=(\mathbf{x}^T\mathbf{x}+\lambda\mathbf{I})\tilde{\beta}_\lambda$$

and solve:

$$\tilde{\beta}_\lambda=(\mathbf{x}^T\mathbf{x}+\lambda\mathbf{I})^{-1}\mathbf{x}^T\mathbf{y}$$

Notice what we've done here: we've taken the old matrix $\mathbf{x}^T\mathbf{x}$ and we've added λ to every one of its diagonal entries. (This is the "ridge" that gives ridge regression its name.) If the predictor variables were centered, this would amount to estimating the coefficients as though each of them as had a little bit more variance than they really did, while leaving all the covariances alone. This would break any exact multicollinearity, so the inverse always exists, and there is always some solution.

What about the intercept? The intercept is different from the other coefficients; it's just a fudge factor we put in to make sure that the regression line goes through the mean of the data. It doesn't make as much sense to penalize its length, so ridge regression is usually done after centering all the variables, both the predictors and the response. This doesn't change the slopes, but sets the intercept to zero. Then, after we have $\tilde{\beta}_\lambda$, we get the intercept by plugging it in to $\overline{y}=\overline{x}\tilde{\beta}_\lambda$.

There are two prices to doing this.

1. We need to pick λ (equivalently, c) somehow.

2. Our estimates of the coefficients are no longer unbiased, but are "shrunk" towards zero.

Point (2) is not as bad as it might appear. If λ is fixed, and we believe our modeling assumptions, we can calculate the bias and variance of the ridge estimates:

$$\begin{aligned}\mathbb{E}\left[\tilde{\beta}_\lambda\right] &= \left(\mathbf{x}^T\mathbf{x}+\lambda\mathbf{I}\right)^{-1}\mathbf{x}^T\mathbb{E}[\mathbf{Y}] &(15.23)\\ &= \left(\mathbf{x}^T\mathbf{x}+\lambda\mathbf{I}\right)^{-1}\mathbf{x}^T\mathbf{x}\beta &(15.24)\end{aligned}$$

[8] $\lambda = 0$ means we ignore the length of the coefficient vector and we're back to ordinary least squares. $\lambda < 0$ would mean we *prefer* larger coefficients, and would lead to some truly perverse consequences.

$$\begin{aligned}
\mathrm{Var}\left[\tilde{\beta}_\lambda\right] &= \mathrm{Var}\left[\left(\mathbf{x}^T\mathbf{x}+\lambda\mathbf{I}\right)^{-1}\mathbf{x}^T\mathbf{Y}\right] && (15.25)\\
&= \mathrm{Var}\left[\left(\mathbf{x}^T\mathbf{x}+\lambda\mathbf{I}\right)^{-1}\mathbf{x}^T\epsilon\right] && (15.26)\\
&= \left(\mathbf{x}^T\mathbf{x}+\lambda\mathbf{I}\right)^{-1}\mathbf{x}^T\sigma^2\mathbf{I}\mathbf{x}\left(\mathbf{x}^T\mathbf{x}+\lambda\mathbf{I}\right)^{-1} && (15.27)\\
&= \sigma^2\left(\mathbf{x}^T\mathbf{x}+\lambda\mathbf{I}\right)^{-1}\mathbf{x}^T\mathbf{x}\left(\mathbf{x}^T\mathbf{x}+\lambda\mathbf{I}\right)^{-1} && (15.28)
\end{aligned}$$

Notice how both of these expressions smoothly approach the corresponding formulas ones for ordinary least squares as $\lambda \to 0$. Indeed, under the Gaussian noise assumption, $\tilde{\beta}_\lambda$ actually has a Gaussian distribution with the given expectation and variance.

Of course, if λ is not fixed in advance, we'd have to worry about the randomness in the distribution of λ. A common practice here is **data splitting**: randomly divide the data into two parts, and use one to pick λ and the other to then actually estimate the parameters, which will have the stated bias and standard errors. (Typically, but not necessarily, the two parts of the data are equally big.)

As for point (1), picking λ, this is also a solvable problem. The usual approach is **cross-validation**: trying a lot of different values of λ, estimate the model on all but one data point, and then see how well different λ's predict that held-out data point. Since there's nothing special about one data point rather than another, do this for each data point, and average the out-of-sample squared errors. Pick the λ which does best at predicting data it didn't get to see. (There are lots of variants, some of which we'll cover later in the course.)

15.4.1 Some Words of Advice about Ridge Regression

Units and Standardization If the different predictor variables don't have physically comparable units[9], it's a good idea to standardize them first, so they all have mean 0 and variance 1. Otherwise, penalizing $\beta^T\beta = \sum_{i=1}^{p}\beta_i^2$ seems to be adding up apples, oranges, and the occasional bout of regret. (Some people like to pre-standardize even physically comparable predictors.)

Stabilization I've presented ridge regression as a way of dealing with multicollinearity, which it is, but it's also perfectly possible to use it when that isn't an issue. The goal there is to **stabilize** the estimates — to reduce their variance, at the cost of a bit of bias. If the linear model is perfectly well-specified, there's little point to doing this, but it can often improve predictions a lot when the model is mis-specified.

15.4.2 Penalties vs. Constraints

I explained ridge regression above as applying a penalty to long coefficient vectors. There is an alternative perspective which is mathematically equivalent, where instead we *constrain* the length of the coefficient vector.

[9]E.g., if they're all masses expressed in grams, they're comparable; if some masses are in kilograms or pounds, they're not comparable but they could easily be made so; if some of them are lengths or prices, they're not physically comparable no matter what units you use.

To see how this works, let's start by setting up the problem: pick some $c > 0$, and then ask for

$$\min_{b \,:\, ||b|| \leq c} \frac{1}{n}(\mathbf{y}-\mathbf{x}\mathbf{b})^T(\mathbf{y}-\mathbf{x}\mathbf{b})$$

Since $||b|| \leq c$ if and only if $||b||^2 \leq c^2$, we might as well say

$$\min_{b \,:\, ||b||^2 \leq c^2} \frac{1}{n}(\mathbf{y}-\mathbf{x}\mathbf{b})^T(\mathbf{y}-\mathbf{x}\mathbf{b})$$

At this point, we invoke the magic of Lagrange multipliers[10]: we can turn a constrained problem into an unconstrained problem with an additional term, and an additional variable:

$$\min_{\mathbf{b},\lambda} \frac{1}{n}(\mathbf{y}-\mathbf{x}\mathbf{b})^T(\mathbf{y}-\mathbf{x}\mathbf{b}) + \lambda(\mathbf{b}^T\mathbf{b} - c^2)$$

Minimizing over λ means that either $\lambda = 0$, or $\mathbf{b}^T\mathbf{b} = c^2$. The former situation will apply when the unconstrained minimum is within the ball $||b|| \leq c$; otherwise, the constraint will "bite", and λ will take a non-zero value to enforce it. As c grows, the required constraint λ will become smaller[11].

When we minimize over $\mathbf{b}$, the precise value of c^2 doesn't matter; only λ does. If we know λ, then we are effectively just solving the problem

$$\min_{\mathbf{b}} \frac{1}{n}(\mathbf{y}-\mathbf{x}\mathbf{b})^T(\mathbf{y}-\mathbf{x}\mathbf{b}) + \lambda\mathbf{b}^T\mathbf{b}$$

which is the penalized regression problem we solved before.

15.4.3 Ridge Regression in R

There are several R implementations of ridge regression; the `MASS` package contains one, `lm.ridge`, which needs you to specify λ. The `ridge` package (Cule, 2014) has `linearRidge`, which gives you the option to set λ, or to select it automatically via cross-validation. (See the next section for a demo of this in action.) There are probably others I'm not aware of.

15.4.4 Other Penalties/Constraints

Ridge regression penalizes the mean squared error with $||b||^2$, the squared length of the coefficient vector. This suggests the idea of using some other measure of how big the vector is, some other **norm**. A mathematically popular family of norms are the ℓ_q norms[12], defined as

$$||b||_q = \left(\sum_{i=1}^{p} |b_i|^q\right)^{1/q}$$

[10]See further reading, if you have forgotten about Lagrange multipliers.

[11]In economic terms, λ is the internal or "shadow" price we'd pay, in units of MSE, to loosen the constraint.

[12]Actually, people usually call them the ℓ_p norms, but we're using p for the number of predictor variables.

The usual Euclidean length is ℓ_2, while ℓ_1 is

$$||b||_1 = \sum_{i=1}^{p} |b_i|$$

and (by continuity $||b||_0$ is just the number of non-zero entries in b. When $q \neq 2$, penalizing the $||b||_q$ does not, usually, have a nice closed-form solution like ridge regression does. Finding the minimum of the mean squared error under an ℓ_1 penalty is called **lasso regression** or the **lasso estimator**, or just **the lasso**[13]. This has the nice property that it tends to give *sparse* solutions — it sets coefficients to be exactly zero (unlike ridge). There are no closed forms for the lasso, but there are efficient numerical algorithms. Penalizing ℓ_0, the number of non-zero coefficients, *sounds* like a good idea, but there are, provably, no algorithms which work substantially faster than trying all possible combinations of variables.

15.5 High-Dimensional Regression

One situation where we know that we will always have multicollinearity is when $n < p$. After all, n points always define a linear subspace of (at most) $n-1$ dimensions[14]. When the number of predictors we measure for each data point is bigger than the number of data points, the predictors *have* to be collinear, indeed multicollinear. We are then said to be in a **high-dimensional** regime.

This is an increasingly common situation in data analysis. A very large genetic study might sequence the genes of, say, 500 people — but measure 500,000 genetic markers in each person[15]. If we want to predict some characteristic of the people from the genes (say their height, or blood pressure, or how quickly they would reject a transplanted organ), there is simply no way to estimate a model by ordinary least squares. Any approach to high-dimensional regression *must* involve either reducing the number of dimensions, until it's $< n$ (as in principle components regression), or penalizing the estimates to make them stable and regular (as in ridge regression), or both.

There is a bit of a myth in recent years that "big data" will solve all our problems, by letting us make automatic predictions about everything without any need for deep understanding. The truth is almost precisely the opposite: when we can measure everything about everyone, p/n blows up, and we are in desperate need of ways of filtering the data and/or penalizing our models. Blindly relying on generic methods of dimension reduction or penalization is going to impose all sorts of bizarre biases, and will work much worse than *intelligent* dimension reduction and *appropriate* penalties, based on actual understanding.

[13]Officially, "lasso" here is an acronym for "least angle selection and shrinkage operator". If you believe that phrase came before the acronym, I would like your help in getting some money out of Afghanistan.

[14]Two points define a line, unless the points coincide; three points define a plane, unless the points fall on a line; etc.

[15]I take these numbers, after rounding, from an actual study done in the CMU statistics department a few years ago.

15.5.1 Demo

Let's apply ridge regression to the simulated data already created, where one predictor variable (X_3) is just the average of two others (X_1 and X_2).

```
library(ridge)
# Fit a ridge regression lambda='automatic' is actually the default setting
demo.ridge <- linearRidge(y ~ x1 + x2 + x3 + x4, data = df, lambda = "automatic")
coefficients(demo.ridge)
## (Intercept)          x1          x2          x3          x4
## 0.629993053 0.463013563 0.052073523 0.474855858 0.006854626
demo.ridge$lambda
## [1] 0.005977451 0.007744569 0.004531840
```

We may compare the predictions we get from this to the predictions we'd from dropping, say, X_2 (Figure 15.6). One substantial advantage of ridge regression is that we don't have to make any decisions about which variables to remove, but can match (to extremely high accuracy) what we'd get after dropping variables.

15.6 Further Reading

Ridge regression, by that name, goes back to Hoerl and Kennard (1970). Essentially the same idea was introduced some years earlier by the great Soviet mathematician A. N. Tikhonov in a series of papers about "regularizing ill-posed optimization problems", i.e., adding penalties to optimization problems to make a solution unique, or to make the solution more stable. For this reason, ridge regression is sometimes also called "Tikhonov regularization" of linear least squares[16].

The use of principal components as a technique of dimension reduction goes back at least to Hotelling in the 1930s, or arguably to Karl Pearson around 1900. I have not been able to trace who first suggested regressing a response variable on the principal components of the predictors. Dhillon *et al.* (2013) establishes a surprising connection between regression on the principal components and ridge regression.

On the use of Lagrange multipliers to enforce constraints on optimization problems, and the general equivalence between penalized and constrained optimization, see Klein (2001), or Shalizi (forthcoming, §E.3).

For high-dimensional regression in general, the opening chapters of Bühlmann and van de Geer (2011) are very good. Bühlmann (2014); Wainwright (2014) may be more accessible review articles on basically the same topics.

For a representative example of the idea that big data "makes theory obsolete", see Anderson (2008); for a reply from someone who actually knows what they're talking about, see `http://earningmyturns.blogspot.com/2008/06/end-of-theory-data-deluge-makes.html`.

[16]There are a large number of variant transliterations of "Tikhonov".

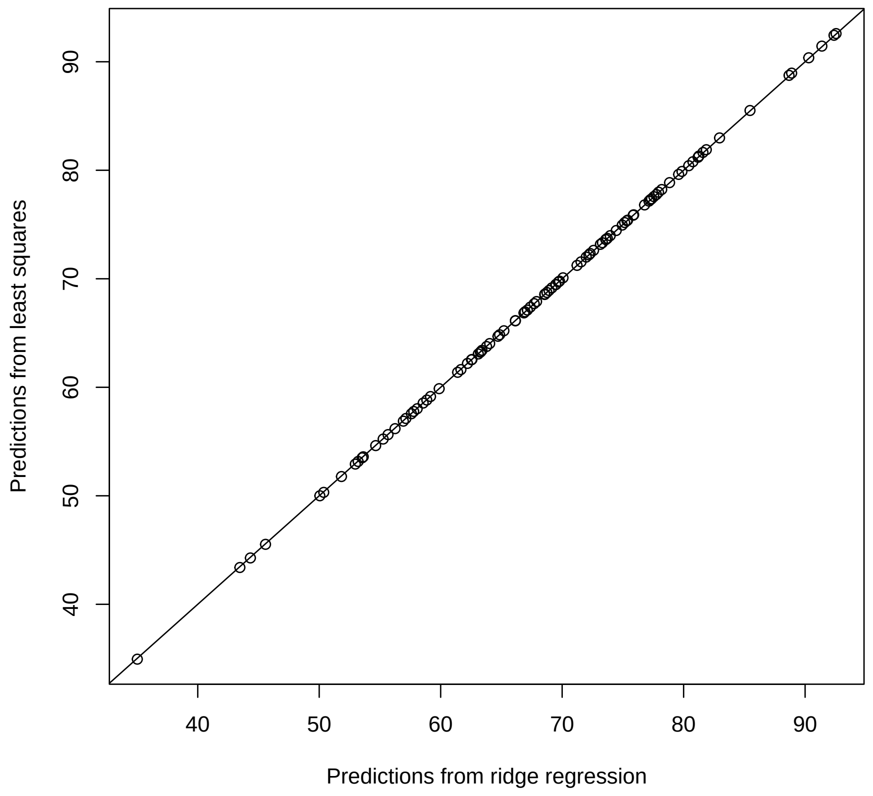


FIGURE 15.6: *Comparison of fitted values from an ordinary least squares regression where we drop X_2 from our running example (vertical axis) against fitted values from a ridge regression on all variables (horizontal axis); the two sets of numbers are not* exactly *equal, though they are close.*

Chapter 16

Tests and Confidence Sets for Multiple Coefficients, Assuming Gaussian Noise

16.1 Assumptions, and Summary of Previous Results

Throughout this chapter, we'll assume that the Gaussian-noise multiple linear regression model

$$Y = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p + \epsilon \tag{16.1}$$

with $\epsilon \sim N(0, \sigma^2)$ independent of the X_is and independent across observations, is completely correct. We will also use the least squares or maximum likelihood estimates of the slopes,

$$\widehat{\beta} = (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{y} \tag{16.2}$$

Under these assumptions, the estimator has a multivariate Gaussian distribution,

$$\widehat{\beta} \sim MVN(\beta, \sigma^2(\mathbf{x}^T\mathbf{x})^{-1}) \tag{16.3}$$

The maximum likelihood estimate of σ^2, $\hat{\sigma}^2$, is

$$\hat{\sigma}^2 = \frac{1}{n}(\mathbf{y} - \mathbf{x}\widehat{\beta})^T(\mathbf{y} - \mathbf{x}\widehat{\beta}) \tag{16.4}$$

This is slightly negatively biased, $\mathbb{E}\left[\hat{\sigma}^2\right] = \frac{n-p-1}{n}\sigma^2$, and has the sampling distribution

$$\frac{n\hat{\sigma}^2}{\sigma^2} \sim \chi^2_{n-p-1} \tag{16.5}$$

$\hat{\sigma}^2\frac{n}{n-p-1}$ is an unbiased estimator of σ^2.

16.2 $z-$ and $t-$ Tests for Single Coefficients

Let's write the true standard error of the estimator $\hat{\beta}_i$ as $\mathrm{se}\left[\hat{\beta}_i\right]$. From the general theory about the variance of $\widehat{\beta}$,

$$\mathrm{se}\left[\hat{\beta}_i\right] = \sqrt{\sigma^2(\mathbf{x}^T\mathbf{x})^{-1}_{i+1,i+1}} \tag{16.6}$$

(Why $i+1$?) Further, from the Gaussian distribution of $\widehat{\beta}$,

$$\frac{\hat{\beta}_i - \beta_i}{\mathrm{se}\left[\hat{\beta}_i\right]} \sim N(0,1) \tag{16.7}$$

If we know σ^2, so that we can compute $\mathrm{se}\left[\hat{\beta}_i\right]$, we can use this to either test hypotheses about the exact value of β_i, or to form confidence intervals. Specifically, a $1-\alpha$ CI would be

$$\hat{\beta}_i \pm z(\alpha/2)\mathrm{se}\left[\hat{\beta}_i\right] \tag{16.8}$$

with z_p being the p^{th} quantile of the standard Gaussian distribution.

If we use instead the unbiased estimate of σ^2, $\hat{\sigma}^2\frac{n}{n-p-1}$, to obtain an estimate $\widehat{\mathrm{se}}\left[\hat{\beta}_i\right]$, we find rather

$$\frac{\hat{\beta}_i - \beta_i}{\widehat{\mathrm{se}}\left[\hat{\beta}_i\right]} \sim t_{n-p-1} \tag{16.9}$$

The reasoning for this is exactly parallel to why we got t_{n-2} distributions for simple linear regression, so I won't rehearse it here. It follows that

$$\hat{\beta}_i \pm t_{n-p-1}(\alpha/2)\widehat{\mathrm{se}}\left[\hat{\beta}_i\right] \tag{16.10}$$

is a $1-\alpha$ confidence interval for β_i. This is implemented in the `confint` function, when applied to the output of `lm`.

As $n \to \infty$, this becomes

$$\hat{\beta}_i \pm z(\alpha/2)\hat{\sigma}\sqrt{(\mathbf{x}^T\mathbf{x})^{-1}_{i+1,i+1}} \tag{16.11}$$

which is often a quite practical alternative to the t-based interval.

16.2.1 What, Exactly, Is `summary` Testing?

When you run `summary` on the output of `lm`, part of what it delivers is a table containing estimated coefficients and standard errors, along with a t-statistic and a p-value for each one. It is important to be very clear about what is being tested here. There is in fact a *different* null hypothesis for each row of the table. The null hypothesis for β_i is that

$$Y = \beta_0 + \beta_1 X_1 + \ldots \beta_{i-1} X_{i-} + 0 X_i + \beta_{i+1} X_i + \ldots + \beta_p X_p + \epsilon \tag{16.12}$$

with ϵ being mean-zero, constant-variance independent Gaussian noise. The alternative hypothesis is that

$$Y = \beta_0 + \beta_1 X_1 + \ldots \beta_{i-1} X_{i-} + \beta_i X_i + \beta_{i+1} X_i + \ldots + \beta_p X_p + \epsilon \tag{16.13}$$

with $\beta_i \neq 0$, and the same assumptions about ϵ. This matters because *whether the null hypothesis is true or not depends on what other variables are included in the model.* The optimal coefficient on X_i might be zero with one set of covariates and non-zero with another. The t test is, by its nature, incapable of saying whether X_i should be included in the model or not.

(This is in addition to the usual cautions about whether testing $\beta_i = 0$ is really informative, about not mistaking "detectably different from zero" for "important", and about how any $\beta_i \neq 0$ will eventually have a p-value arbitrarily close to 0.)

16.2.2 No, Really, Whether Coefficients Are Zero Changes with the Covariates

Here is the simplest situation I know of which illustrates that the true (optimal or population-level) coefficient of a given predictor variable changes with the other variables included in the model. Suppose that the true model is

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon \tag{16.14}$$

with all the usual assumptions being met. Without knowing this, we instead estimate the model

$$Y = \gamma_0 + \gamma_1 X_1 + \eta \tag{16.15}$$

We know, from our study of the simple linear model, that the (optimal or population) value of γ_1 is

$$\gamma_1 = \frac{\mathrm{Cov}[X_1, Y]}{\mathrm{Var}[X_1]} \tag{16.16}$$

Substituting in for Y,

$$\begin{aligned}
\gamma_1 &= \frac{\mathrm{Cov}[X_1, \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon]}{\mathrm{Var}[X_1]} && (16.17)\\
&= \frac{\mathrm{Cov}[X_1, \beta_0] + \mathrm{Cov}[X_1, \beta_1 X_1] + \mathrm{Cov}[X_1, \beta_2 X_2] + \mathrm{Cov}[X_1, \epsilon]}{\mathrm{Var}[X_1]} && (16.18)\\
&= \frac{0 + \beta_1 \mathrm{Cov}[X_1, X_1] + \beta_2 \mathrm{Cov}[X_1, X_2] + 0}{\mathrm{Var}[X_1]} && (16.19)\\
&= \beta_1 + \beta_2 \frac{\mathrm{Cov}[X_1, X_2]}{\mathrm{Var}[X_1]} && (16.20)
\end{aligned}$$

Thus, even if $\beta_1 = 0$, we can easily have $\gamma_1 \neq 0$, and vice versa. (See also Exercise 1.)

16.3 Variance Ratio (F) Tests for Multiple Coefficients Being Zero

If we want to test whether a group of multiple coefficients are all simultaneously zero, the traditional approach is a variance ratio or F test. To lay everything out, the null hypothesis is that

$$Y = \beta_0 + \beta_1 X_1 + \ldots + \beta_q X_q + 0X_{q+1} + \ldots + 0X_p + \epsilon \tag{16.21}$$

while the alternative is

$$Y = \beta_0 + \beta_1 X_1 + \ldots + \beta_q X_q + \beta_{q+1} X_{q+1} + \ldots + \beta_p X_p + \epsilon \tag{16.22}$$

with at least one of the coefficients $\beta_{q+1}, \ldots \beta_p \neq 0$. The null hypothesis, then, is that in a linear model which includes all the predictors $X_1, \ldots X_p$, the optimal coefficients for the last $p-q$ variables are all zero.

For both models, we get an estimate of σ^2, say $\hat{\sigma}^2_{null}$ for the null model (with coefficients fixed at zero) and $\hat{\sigma}^2_{full}$ for the full model. Because the null model is a special case of the full model, and we estimate parameters in each case by minimizing the MSE, $\hat{\sigma}^2_{null} \geq \hat{\sigma}^2_{full}$.

Following reasoning exactly parallel to the way we got the F test for the simple linear regression model (Chapter 10),

$$\frac{n\hat{\sigma}^2_{full}}{\sigma^2} \sim \chi^2_{n-p-1} \tag{16.23}$$

while, under the null hypothesis,

$$\frac{n(\hat{\sigma}^2_{null} - \hat{\sigma}^2_{full})}{\sigma^2} \sim \chi^2_{p-q} \tag{16.24}$$

and so (again under the null hypothesis)

$$\frac{(\hat{\sigma}^2_{null} - \hat{\sigma}^2_{full})/(p-q)}{\hat{\sigma}^2_{full}/(n-p-1)} \sim F_{p-q,n-p-1} \tag{16.25}$$

We therefore reject the null hypothesis when the test statistic

$$F = \frac{(\hat{\sigma}^2_{null} - \hat{\sigma}^2_{full})/(p-q)}{\hat{\sigma}^2_{full}/(n-p-1)} \tag{16.26}$$

is too large compared to the $F_{p-q,n-p-1}$ distribution. This is why this is called an F test for this set of regression coefficients. If we're not testing all the coefficients at once, this is a **partial** F test.

The proper interpretation of this test is "Does letting the slopes for $X_{q+1},\ldots X_p$ be non-zero reduce the MSE more than we would expect just by noise?" As n grows, increasingly small improvements will become clearly detectable as not-noise, so increasingly small but non-zero sets of coefficients will be detected as significant by the F test.

Cautions The variance ratio test does not test any of the following:

- Whether some variable not among $X_1,\ldots X_p$ ought to be included in the model.
- Whether the relationship between Y and the X_i is linear.
- Whether the Gaussian noise assumption holds.
- Whether any of the other modeling assumptions hold.

16.3.1 All Slopes at Once

An obvious special case is the hypothesis that all the coefficients are zero. That is, the null hypothesis is

$$Y = \beta_0 + 0X_1 + \ldots + 0X_p + \epsilon \tag{16.27}$$

with the alternative being the full model

$$Y = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p + \epsilon \tag{16.28}$$

The estimate of σ^2 under the null is the sample variance of Y, s^2_Y, so the test statistic becomes

$$\frac{(s^2_Y - \hat{\sigma}^2_{full})/p}{\hat{\sigma}^2_{full}/(n-p-1)} \tag{16.29}$$

whose distribution under the null is $F_{p,n-p-1}$.

This **full** F test is often called a test of the significance of the whole regression. This is true, but has to be understood in a very specific sense. We are testing whether, if Y is linearly regressed on $X_1,\ldots X_p$ and only on those variables, the reduction in the MSE from actually estimating slopes over just using a flat regression surface is bigger than we'd expect from pure noise. Once again, the test has no power to detect violations of any of the modeling assumptions. (See the discussion of the F test for simple linear regression in Chapter 10.)

16.3.2 Variance Ratio Tests in R

This is most easily done through the `anova` function. We fit the null model and the full model, both with `lm`, and then pass them to the `anova` function:

```
mobility <- read.csv("http://www.stat.cmu.edu/~cshalizi/mreg/15/dap/1/mobility.csv")
mob.null <- lm(Mobility ~ Commute, data = mobility)
mob.full <- lm(Mobility ~ Commute + Latitude + Longitude, data = mobility)
anova(mob.null, mob.full)
## Analysis of Variance Table
##
## Model 1: Mobility ~ Commute
## Model 2: Mobility ~ Commute + Latitude + Longitude
##   Res.Df    RSS Df Sum of Sq      F   Pr(>F)
## 1    727 1.3143
## 2    725 1.2952  2  0.019111 5.3491 0.004942
```

The second row tells us that the full model has two more parameters than the null, that $n(\hat{\sigma}^2_{null} - \hat{\sigma}^2_{full}) = 0.0191114$, and then what the variance ratio or F statistic and the corresponding p-value are. Here, we learn that the decrease in the root-MSE which comes from adding latitude and longitude as predictors, while very small (0.51 percentage points) is large enough that it is unlikely to have arisen by capitalizing on noise[1].

16.3.3 Variable Deletion via F Tests

It's not uncommon to use F tests for variable deletion: pick your least favorite set of predictors, test whether all of their βs are zero, and, if so, delete them from the model (and re-estimate). Presuming that we can trust the modeling assumptions, there are still a few points about this procedure which are slightly dubious, or at least call for much more caution than is often exercised.

Statistical power The test controls the probability of rejecting when the null is true — it guarantees that if $\beta_q = 0$, we have a low probability of rejecting that null hypothesis. For deletion to be reliable, however, we'd want a low probability of *missing* variables with non-zero coefficients, i.e., a low probability of retaining the null hypothesis when it's wrong, or high power to detect departures from the null. Power cannot be read off from the p-value, and grows with the magnitude of the departure from the null. One way to get at this is, as usual, to complement the hypothesis test with a confidence set for the coefficients in question. Ignoring variables whose coefficients are *precisely* estimated to be close to zero is much more sensible than ignoring variables because their coefficients can only be estimated very loosely.

Non-transivitiy The variance ratio test checks whether the MSE of the smaller model is significantly or detectably worse than the MSE of the full model. One drawback to this is that a series of insignificant, undetectably-small steps can add up to a significant, detectably-big change. In mathematical jargon: "is equal to" is a transitive

[1]Once again, this presumes that the only two possibilities in the world are a completely-correct linear-Gaussian model with just commuting time as a predictor, and a completely-correct linear-Gaussian model with commuting time, latitude and longitude as predictors.

relation, so that if $A = B$ and $B = C$, $A = C$. But "insignificantly different from" is not a transitive relation, so if $A \approx B$ and $B \approx C$, we can't conclude $A \approx C$.

Concretely: a group of variables might show up as significant in a partial F test, even though none of them was individually significant on a t test in the full model[2]. Also, if we delete variables in stages, we can have a situation where at each stage the increase in MSE is insignificant, but the difference between the full model and the final model is highly significant.

16.3.4 Likelihood Ratio Tests

As with the F test for simple linear models, there is an alternative based on the likelihood ratio. As with the simple model (Chapter 10), the log-likelihood of the model, at the maximum likelihood estimate, is

$$-\frac{n}{2}(1+\log 2\pi) - \frac{n}{2}\log \hat{\sigma}^2 \tag{16.30}$$

Hence the difference in log-likelihoods between the full model, with all p slopes estimated, and the null model, with only q slopes estimated and the other $p-q$ fixed, is

$$\Lambda = -\frac{n}{2}\log \hat{\sigma}^2_{full} + \frac{n}{2}\log \hat{\sigma}^2_{null} = \frac{n}{2}\log \frac{\hat{\sigma}^2_{null}}{\hat{\sigma}^2_{full}} \tag{16.31}$$

This is the log of the ratio of likelihoods (not the ratio of log likelihoods!) Under the null hypothesis[3],

$$2\Lambda \sim \chi^2_{p-q} \tag{16.32}$$

The same cautions apply to the likelihood ratio test as to the F test: it does not check modeling assumptions.

One advantage of likelihood ratio tests is that exactly the same procedure can be used to test the hypothesis that $\beta_q = \mathbf{0}$ and to test $\beta_q = \beta^*_q$, for any other particular vector of parameters. For that matter, it can be used to test $\mathbf{c}\beta = \mathbf{r}$, where $\mathbf{c}$ is any non-random $q \times (p+1)$ matrix, and $\mathbf{r}$ is any non-random $q \times 1$ vector. Thus, for example, it can be used to test the hypothesis that two slopes are *equal*, or that all slopes are equal, etc.

Likelihood Ratio vs. F Tests For linear-Gaussian models, both the likelihood ratio and the F statistic are functions of the ratio $\hat{\sigma}^2_{null}/\hat{\sigma}^2_{full}$ (Exercise 2). For fixed p and q, as $n \rightarrow \infty$, the two tests deliver the same p-values when $\hat{\sigma}^2_{null}/\hat{\sigma}^2_{full}$ is the same. At finite n, they are somewhat different, with the F test usually giving a somewhat higher p value than the χ^2 test, particularly if p is close to n. Which test is more *accurate* is another question. The likelihood ratio test can actually work for large n when the

[2]This is yet another reason not to pay much attention to the p-values reported by `summary`.

[3]Strictly speaking, this only becomes exact as $n \rightarrow \infty$. This issue is that deriving the χ^2 distribution for Λ presumes every parameter's maximum likelihood estimate has a Gaussian distribution around its true value (see Chapter 10), and while this is true for the $\hat{\beta}_i$s, it is only approximately true for $\hat{\sigma}^2$. See Exercise 4.

model is mis-specified, in the sense of telling us which wrong model is closer to the truth (Vuong, 1989), while the F test's refinements over the χ^2 very much depend on all the modeling assumptions being right.

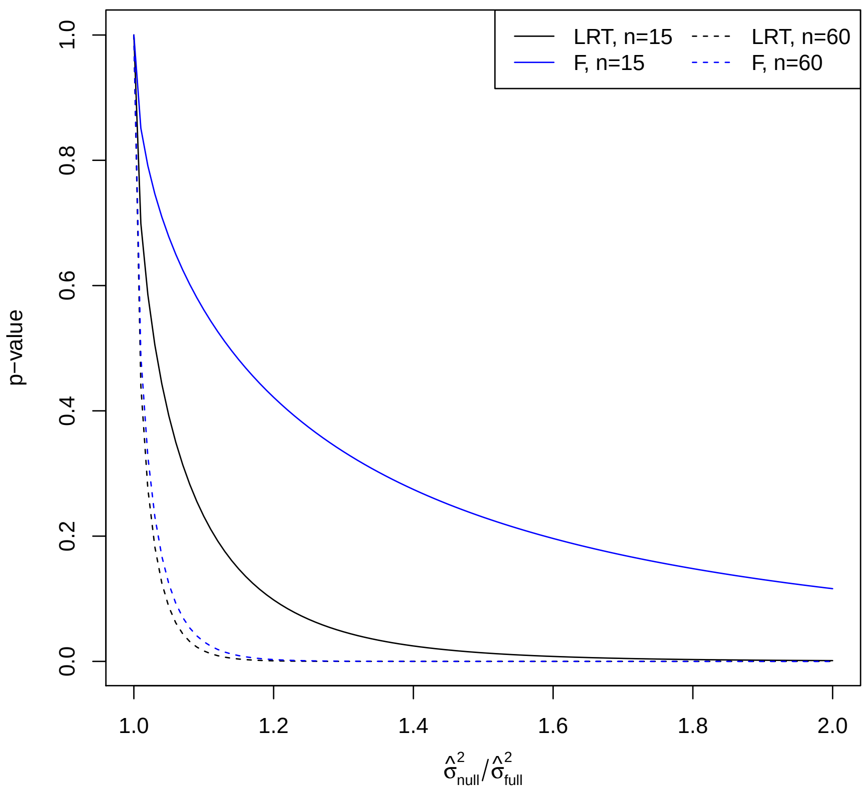


FIGURE 16.1: *Difference in p-values obtained from using a likelihood ratio test (black) and an F test (blue) on the same data, with $p = 10$, $q = 9$, and n either 15 (solid) or 60 (dotted). In general, the difference between the two tests goes to zero as $n - p$ grows. (See next page for code.)*

```
n <- 15
p <- 10
q <- 9
curve(pchisq(n * log(x), df = p - q, lower.tail = FALSE), from = 1, to = 2,
    xlab = expression(hat(sigma)[null]^2/hat(sigma)[full]^2), ylab = "p-value")
curve(pf((x - 1) * (n - p - 1)/(p - q), p - q, n - p - 1, lower.tail = FALSE),
    add = TRUE, col = "blue")
n2 <- 60
curve(pchisq(n2 * log(x), df = p - q, lower.tail = FALSE), add = TRUE, lty = "dashed")
curve(pf((x - 1) * (n2 - p - 1)/(p - q), p - q, n2 - p - 1, lower.tail = FALSE),
    add = TRUE, col = "blue", lty = "dashed")
legend("topright", ncol = 2, legend = c("LRT, n=15", "F, n=15", "LRT, n=60",
    "F, n=60"), col = c("black", "blue", "black", "blue"), lty = c("solid",
    "solid", "dashed", "dashed"))
```

FIGURE 16.2: *Code for generating Figure 16.1.*

16.4 Confidence Sets for Multiple Coefficients

Suppose we want to do inference on two coefficients, say β_i and β_j, at once. That means we need to come up with a two-dimensional confidence region $C(\alpha)$, where we can say that $\mathbb{P}\left((\beta_i, \beta_j) \in C(\alpha)\right) = 1-\alpha$. This would involve the same sort of trilemma as confidence intervals for single coefficients. That is, one of three things must be true:

1. Both β_i and β_j are in $C(\alpha)$; or
2. We got data which was very ($\leq \alpha$) improbable under all possible values of the parameters; or
3. Our model is wrong.

If we trust our model, then, we can indeed be confident that both β_i and β_j are simultaneously in $C(\alpha)$.

Clearly, nothing depends on wanting to do inference on just two coefficients at once; we could consider any subset of them we like, up to all $p+1$ of them.

With one parameter, intervals are the most natural confidence sets to work with. With more than one parameter, we have choices to make about the *shape* of the confidence set. The two easiest ones to work with are rectangular boxes, and ellipsoids.

16.4.1 Confidence Boxes or Rectangles

The natural thing to want to do is to take a confidence interval for each coefficient and put them together into a confidence box or rectangle. For instance, using the t-distribution CI for β_i and β_j, the box would be

$$(\hat{\beta}_i \pm t_{n-p-1}(\alpha/2)\widehat{\mathrm{se}}\left[\hat{\beta}_i\right]) \times (\hat{\beta}_j \pm t_{n-p-1}(\alpha/2)\widehat{\mathrm{se}}\left[\hat{\beta}_j\right]) \tag{16.33}$$

(And similarly for three or more parameters.) This is, however, not quite right as I've written it. The problem is that while each interval covers its true coefficient with high probability, both intervals *simultaneously* cover the pair of parameters is a different story. Let me abbreviate the interval for β_i as $C_i(\alpha)$, likewise the interval for β_j is $C_j(\alpha)$. We have

$$\mathbb{P}(\beta_i \in C_i(\alpha)) = 1-\alpha \ , \ \mathbb{P}\left(\beta_j \in C_j(\alpha)\right) = 1-\alpha \tag{16.34}$$

but from this it does not follow that

$$\mathbb{P}\left(\beta_i \in C_i(\alpha), \beta_j \in C_j(\alpha)\right) = 1-\alpha \tag{16.35}$$

To see this, let's consider the complementary event: it's

$$\beta_i \notin C_i(\alpha) \vee \beta_j \notin C_j(\alpha) \tag{16.36}$$

writing $\vee$ for logical-or[4] By basic probability,

$$\mathbb{P}\left(\beta_i \notin C_i(\alpha) \vee \beta_j \notin C_j(\alpha)\right) = \mathbb{P}(\beta_i \notin C_i(\alpha)) + \mathbb{P}\left(\beta_j \notin C_j(\alpha)\right) - \mathbb{P}\left(\beta_i \notin C_i(\alpha), \beta_j \notin C_j(\alpha)\right) \tag{16.37}$$

Since C_i and C_j are $1-\alpha$-confidence sets,

$$\mathbb{P}\left(\beta_i \notin C_i(\alpha) \vee \beta_j \notin C_j(\alpha)\right) = 2\alpha - \mathbb{P}\left(\beta_i \notin C_i(\alpha), \beta_j \notin C_j(\alpha)\right) \leq 2\alpha \tag{16.38}$$

So $C_i(\alpha) \times C_j(\alpha)$ isn't itself a $1-\alpha$ confidence set; its real confidence level could be as little as $1-2\alpha$. If we had been looking at q coefficients at once, the confidence level might have been as low as $1-q\alpha$.

This suggests, however, a very simple, if sometimes over-cautious, way of building a confidence box. If we want the final box to have a $1-\alpha$ confidence level, and we're dealing with q coefficients at once, we calculate $1-\alpha/q$ confidence levels for each coefficient, say $C_i(\alpha/q)$, and then set

$$C(\alpha) = C_1(\alpha/q) \times C_2(\alpha/q) \times \ldots \times C_q(\alpha/q) \tag{16.39}$$

By our reasoning above, this final $C(\alpha)$ will cover all q parameters at once with probability at least $1-\alpha$.

This trick of building a $1-\alpha$ confidence box for q parameters at once from q $1-\alpha/q$ confidence intervals is completely generic; it doesn't just work on regression coefficients, but for any parameters of any statistical model at all. For more on it, see §16.5 below.

[4] That is, $A \vee B$ means in ordinary English "A is true or B is true or both are true".

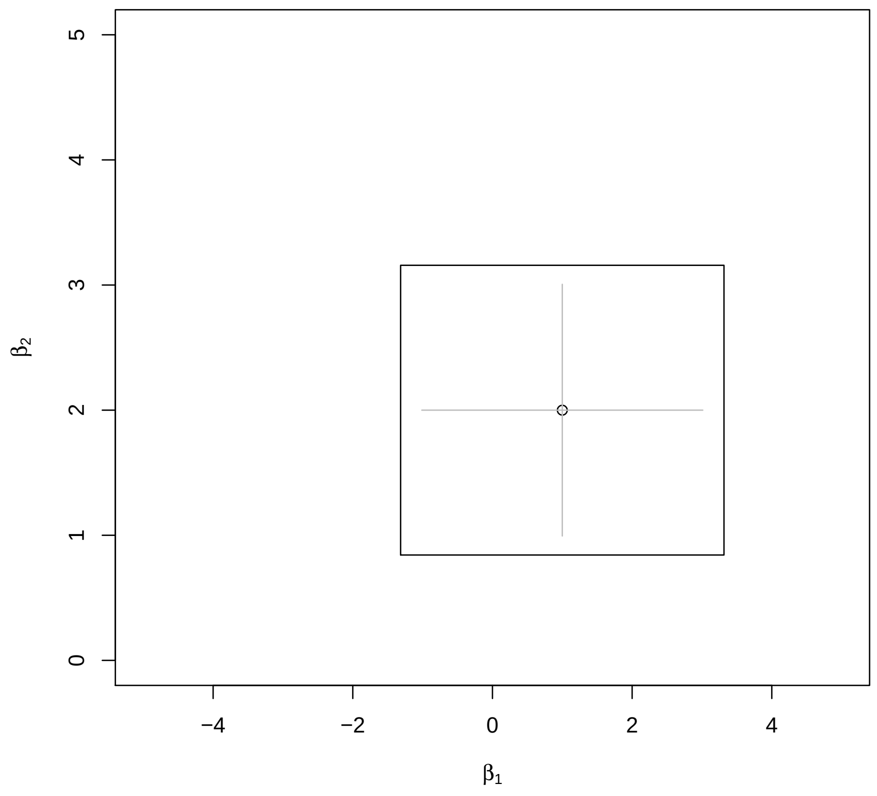


FIGURE 16.3: *Grey lines: 95% confidence intervals for two coefficients, based on inverting t tests, and so centered at the point estimate (dot). Black box: a 95% confidence rectangle for both coefficients simultaneously. Notice that the grey lines do not touch the sides of the rectangle; the latter correspond to 97.5% CIs for each coefficient. If we did draw the rectangle corresponding to the grey lines, its actual confidence level could be as low as 90%. (See source file for code.)*

16.4.2 Confidence Balls or Ellipsoids

An alternative to confidence boxes is to try to make confidence *balls*. To see how this could work, suppose first that $\hat{\beta}_i$ and $\hat{\beta}_j$ were uncorrelated. Since

$$\frac{\hat{\beta}_i - \beta_i}{\mathrm{se}\left[\hat{\beta}_i\right]} \sim N(0,1) \tag{16.40}$$

(and likewise for β_j), we would have[5]

$$\left(\frac{\hat{\beta}_i - \beta_i}{\mathrm{se}\left[\hat{\beta}_i\right]}\right)^2 + \left(\frac{\hat{\beta}_j - \beta_j}{\mathrm{se}\left[\hat{\beta}_j\right]}\right)^2 \sim \chi^2_2 \tag{16.41}$$

Therefore, a simultaneous $1-\alpha$ confidence region for (β_i, β_j) would be the region where

$$\left(\frac{\hat{\beta}_i - \beta_i}{\mathrm{se}\left[\hat{\beta}_i\right]}\right)^2 + \left(\frac{\hat{\beta}_j - \beta_j}{\mathrm{se}\left[\hat{\beta}_j\right]}\right)^2 \leq \chi^2_2(1-\alpha) \tag{16.42}$$

Some geometry shows that this region is an ellipse, its axes parallel to the coordinate axis with the length from end to end along one axis being $2\mathrm{se}\left[\hat{\beta}_i\right]\chi^2_2(1-\alpha)$, and its length along the other axis being $2\mathrm{se}\left[\hat{\beta}_j\right]\chi^2_2(1-\alpha)$.

If we had q different uncorrelated coefficients, the confidence region would be the set $(\beta_1, \beta_2, \ldots \beta_q)$ where

$$\sum_{i=1}^{q} \left(\frac{\hat{\beta}_i - \beta_i}{\mathrm{se}\left[\hat{\beta}_i\right]}\right)^2 \leq \chi^2_q(1-\alpha) \tag{16.43}$$

When $q > 2$, we call this region an "ellipsoid" rather than an "ellipse", but it's the same idea.

Usually, of course, the different coefficient estimates are correlated with each other, so we need to do something a bit different. If we write β_q for the vector of coefficients we're interested in, and $\mathbf{\Sigma}_q$ for its variance-covariance matrix, then the confidence region is the set of all β_q where

$$(\widehat{\beta}_q - \beta_q)^T \mathbf{\Sigma}_q^{-1} (\widehat{\beta}_q - \beta_q) \leq \chi^2_q(1-\alpha) \tag{16.44}$$

This, too, is an ellipsoid, only now the axes point in the directions given by the eigenvectors of $\mathbf{\Sigma}_q$, and the length along each axis is proportional to the square root of the corresponding eigenvalue. (See §16.4.2.2 for a derivation.)

[5]Because when $Z_1, \ldots Z_d$ are independent $N(0,1)$ variables, $\sum_i Z_i^2 \sim \chi^2_d$.

Since $\mathbf{\Sigma}_q$ is a $q \times q$ sub-matrix of $\sigma^2(\mathbf{x}^T\mathbf{x})^{-1}$, we can't actually use this. We can, however, use the appropriate sub-matrix of $\hat{\sigma}^2(\mathbf{x}^T\mathbf{x})^{-1}$ as an approximation, which becomes exact as $n \to \infty$. Similarly, if we use the unbiased estimate of σ^2, we replace $\chi^2_q(1-\alpha)$ with $qF_{q,n-p-1}(1-\alpha)$.

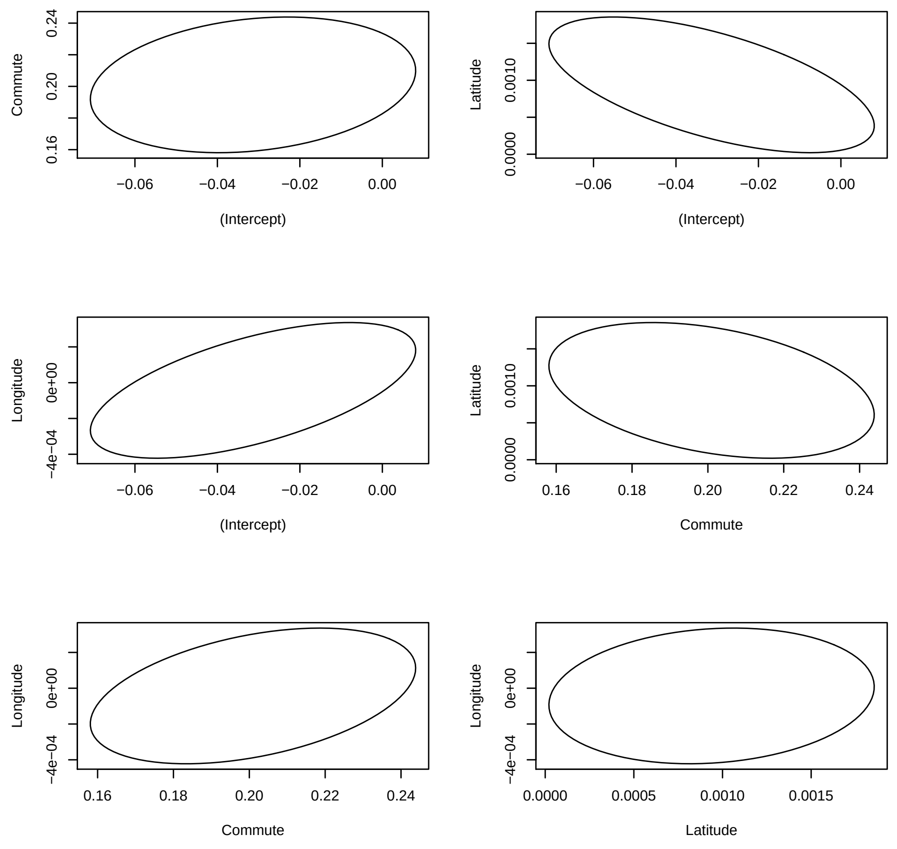
16.4.2.1 Confidence Ellipsoids in R

The package `ellipse` (Murdoch and Chow, 2013) contains functions for plotting 2D confidence ellipses. The main function is also called `ellipse`, which happens to have a specialized method for `lm` models. The usage is

```
my.model <- lm(y ~ x1 + x2 + x3)
plot(ellipse(my.model, which = c(1, 2), level = 0.95))
```

Here `which` is the vector of coefficient indices (it can only be of length 2) and `level` is the confidence level. Notice that what `ellipse` actually returns is a two-column array of coordinates, which can be plotted, or passed along to other graphics functions (like `points` or `lines`). See Figure 16.4.

Three-dimensional confidence ellipsoids can be made with the `rgl` library (Adler *et al.*, 2014). While confidence ellipsoids exist in any number of dimensions, they can't really be visualized when $q > 3$.



```
library(ellipse)
par(mfrow = c(3, 2))
plot(ellipse(mob.full, which = c(1, 2), level = 1 - ((1 - 0.95)/6)), type = "l")
plot(ellipse(mob.full, which = c(1, 3), level = 1 - ((1 - 0.95)/6)), type = "l")
plot(ellipse(mob.full, which = c(1, 4), level = 1 - ((1 - 0.95)/6)), type = "l")
plot(ellipse(mob.full, which = c(2, 3), level = 1 - ((1 - 0.95)/6)), type = "l")
plot(ellipse(mob.full, which = c(2, 4), level = 1 - ((1 - 0.95)/6)), type = "l")
plot(ellipse(mob.full, which = c(3, 4), level = 1 - ((1 - 0.95)/6)), type = "l")
```

FIGURE 16.4: *Confidence ellipses for every pair of coefficients in the model where economic mobility is regressed on the prevalence of short commutes, latitude and longitude. (Remember the intercept is the first coefficient.) Why do I use this odd-looking confidence level?*

16.4.2.2 Where the χ^2_q Comes From

To see why this should be so, we need to do some linear algebra, to turn a Gaussian random vector with correlations and unequal variances into a Gaussian random vector where the coordinates are all $\sim N(0,1)$ and independent of each other. The starting point is the fact that Σ_q is a square, symmetric, positive-definite matrix. Therefore it can be written as follows:

$$\Sigma_q = \mathbf{V}\mathbf{U}\mathbf{V}^T \tag{16.45}$$

where $\mathbf{U}$ is the diagonal matrix of eigenvalues, and $\mathbf{V}$ is the matrix whose columns are the eigenvectors; $\mathbf{V}^T$ is its transpose, and $\mathbf{V}^T\mathbf{V} = \mathbf{I}$. If we define $\Sigma_q^{1/2} = \mathbf{V}\mathbf{U}^{1/2}$, where $\mathbf{U}^{1/2}$ is the diagonal matrix with the square roots of the eigenvalues, then

$$\begin{aligned}
\mathrm{Var}\left[\Sigma_q^{-1/2}(\widehat{\beta}_q - \beta_q)\right] &= \Sigma_q^{-1/2}\mathrm{Var}\left[\widehat{\beta}_q - \beta_q)\right](\Sigma_q^{-1/2})^T & (16.46)\\
&= \mathbf{U}^{-1/2}\mathbf{V}^{-1}\mathbf{V}\mathbf{U}\mathbf{V}^T\mathbf{V}\mathbf{U}^{-1/2} & (16.47)\\
&= \mathbf{U}^{-1/2}\mathbf{U}\mathbf{U}^{-1/2} & (16.48)\\
&= \mathbf{I} & (16.49)
\end{aligned}$$

where the last step works because $\mathbf{U}$ and $\mathbf{U}^{-1/2}$ are both diagonal matrices. In other words, while the coordinates of $\widehat{\beta}_q - \beta_q$ have unequal variances and are correlated with each other, $\Sigma_q^{-1/2}(\widehat{\beta}_q - \beta_q)$ is a random vector where each coordinate has variance 1 and is uncorrelated with the others. Since the initial vector was Gaussian, this too is Gaussian, hence

$$\Sigma_q^{-1/2}(\widehat{\beta}_q - \beta_q) \sim MVN(0, \mathbf{I}) \tag{16.50}$$

Therefore

$$\left(\Sigma_q^{-1/2}(\widehat{\beta}_q - \beta_q)\right)^T \Sigma_q^{-1/2}(\widehat{\beta}_q - \beta_q) \sim \chi^2_q \tag{16.51}$$

since it's a sum of q squared, independent $N(0,1)$ variables.

On the other hand,

$$\begin{aligned}
&\left(\Sigma_q^{-1/2}(\widehat{\beta}_q - \beta_q)\right)^T \left(\Sigma_q^{-1/2}(\widehat{\beta}_q - \beta_q)\right) & (16.52)\\
&= (\widehat{\beta}_q - \beta_q)^T \left(\Sigma_q^{-1/2}\right)^T \Sigma_q^{-1/2}(\widehat{\beta}_q - \beta_q) & \\
&= (\widehat{\beta}_q - \beta_q)^T \mathbf{V}\mathbf{U}^{-1/2}\mathbf{U}^{-1/2}\mathbf{V}^{-1}(\widehat{\beta}_q - \beta_q) & (16.53)\\
&= (\widehat{\beta}_q - \beta_q)^T \mathbf{V}\mathbf{U}^{-1}\mathbf{V}^{-1}(\widehat{\beta}_q - \beta_q) & (16.54)\\
&= (\widehat{\beta}_q - \beta_q)^T \Sigma_q^{-1}(\widehat{\beta}_q - \beta_q) & (16.55)
\end{aligned}$$

Combining Eqs. 16.51 and 16.55,

$$(\widehat{\beta}_q - \beta_q)^T \Sigma_q^{-1}(\widehat{\beta}_q - \beta_q) \sim \chi^2_q \tag{16.56}$$

as was to be shown.

16.5 Further Reading

Variance and likelihood ratio tests go back to the period of the 1910s–1930s; see references in Chapter 10. Further exposition can be found in any textbook on regression, or general mathematical statistics.

The trick in §16.4.1, of getting an over-all confidence level of $1-\alpha$ for q parameters simultaneously, by demanding the higher confidence level of $1-\alpha/q$ for each one separately, is one use of an important tool called **Bonferroni correction** or **Bonferroni adjustment**[6]. For an account of the role of this general idea in probability theory, see Galambos and Simonelli (1996). Bonferroni correction is also often used for hypothesis testing: if we test q distinct hypotheses, and we want to have the probability of making *no* false rejections be α, we can achieve that by having each test be of size α/q. Indeed, we could give each test whatever size we like, so long as the sum of the tests is α.

One sometimes encounters the mis-understanding that Bonferroni correction requires the test statistics or confidence intervals to be statistically independent (e.g., Ashby 2011); as you can see from the argument above, this is just wrong. What is true is that Bonferroni correction is very cautious, and that one can sometimes come up with less conservative ways of doing multiple inference if one either uses more detailed information about how the statistics relate to each other (as in §16.4.2), or one is willing to tolerate a certain number of false positives. The latter idea leads to important work on multiple testing and "false discovery control", which is outside the scope of this course, but see Benjamini and Hochberg (1995); Genovese and Wasserman (2004), and, for an unforgettable demonstration of how ignoring multiple testing issues leads to nonsense, Bennett *et al.* (2010).

16.6 Exercises

1. In the scenario of §16.2.2, is it possible for both ϵ and η to obey the Gaussian noise assumption? That is, it is possible to have $\epsilon \sim N(0,\sigma^2_\epsilon)$, independent of X_1 and X_2, and to have $\eta \sim N(0,\sigma^2_\eta)$, independent of X_1? *Hint:* Suppose X_1 and X_2 are jointly Gaussian, and, for simplicity, that both have mean 0.

2. (a) Show that the variance ratio test statistic (Eq. 16.26) depends on the data only through the ratio $\hat{\sigma}^2_{null}/\hat{\sigma}^2_{full}$.

   (b) Show that as $\hat{\sigma}^2_{null} \rightarrow \hat{\sigma}^2_{full}$,

$$\log\frac{\hat{\sigma}^2_{null}}{\hat{\sigma}^2_{full}} \rightarrow \frac{\hat{\sigma}^2_{null}-\hat{\sigma}^2_{full}}{\hat{\sigma}^2_{full}} \qquad (16.57)$$

3. Chapter 7 argued that every confidence set comes from inverting a hypothesis test. What is the hypothesis test corresponding to the confidence boxes of

[6] Computer scientists, and some mathematicians, call it a "union bound" — can you explain why?

§16.4.1? That is, find an explicit form of the test statistic and of the rejection region.

4. Let $X_n \sim \chi^2_{n-p}$, with fixed p.

   (a) Show that X_n/n approaches a constant a, and find a.

   (b) Show that $(X_n - a)/\sqrt{n}$ approaches a Gaussian distribution, and find the expectation and variance. *Hint:* show that the moment generating functions converge.

   (c) Combine the previous results to write the limiting distribution of X_n/n as a Gaussian, whose parameters (may) change with n.

Chapter 17

Interactions

17.1 Interactions, What Are They?

When we say that there are no interactions between X_i and X_j, we mean that

$$\frac{\partial \mathbb{E}[Y|X=x]}{\partial d x_i} \tag{17.1}$$

is not a function of x_j. Said another way, there are no interactions if and only if

$$\mathbb{E}[Y|X=x] = \alpha + \sum_{i=1}^{p} f_i(x_i) \tag{17.2}$$

so that each coordinate of X makes its own separate, additive contribution to Y. The standard multiple linear regression model of course includes no interactions between any of the predictor variables.

General considerations of probability theory, mathematical modeling, statistical theory, etc., give us no reason whatsoever to anticipate that interactions are rare, or that when they exist they are small. You might be so lucky as to not have any to deal with, but you should not *presume* you will be lucky.

Diagnosing the presence of interactions See Chapter 13 for some ideas about how to do this. One trick not mentioned there is to plot the residuals from an interaction-free model against the product of two predictors, e.g., against X_1X_2. This, however, presumes a particular form for the interaction, gone over in the next section.

17.2 The Conventional Form of Interactions in Linear Models

The usual way of including interactions in a linear model is to add a product term, as, e.g.,

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \epsilon \tag{17.3}$$

Once we add such a term, we estimate β_3 in exactly the same way we'd estimate any other coefficient.

Interpretation In the model of Eq. 17.3, it is no longer correct to interpret β_1 as $\mathbb{E}[Y|X_1 = x_1+1, X_2 = x_2] - \mathbb{E}[Y|X_1 = x_1, X_2 = x_2]$. That difference is, rather $\beta_1 + \beta_3 X_2$. Similarly, β_2 is no longer the expected difference in Y between two otherwise-identical cases where X_2 differs by 1. The fact that we can't give one answer to "how much does the response change when we change this variable?", that the correct answer to that question always involves the other variable, is what interaction *means*.

What we can say is that β_1 is the slope with regard to X_1 when $X_2 = 0$, and likewise β_2 is how much we expect Y to change for a one-unit change in X_2 when $X_1 = 0$. β_3 is the rate at which the slope on X_1 changes as X_2 changes, and likewise the rate at which the slope on X_2 changes with X_1 (see Exercise 1 for why it's both).

Diagnostics and inference Diagnostics for a product term goes just like it would for any other: the residuals should have the same distribution no matter what the value of $X_i X_j$ happens to be; all the usual plots can be made using $X_i X_j$ as the predictor variable. Inference, too, works exactly the same way.

Terminology The coefficients which go with the linear terms, β_1 and β_2 above, are often called the "main effects", while β_3 would be an "interaction effect". I think this terminology is misleading in at least two ways. First, by talking about "effects" at all, it carries causal implications which are not usually warranted by a regression. Second, it implies that the linear terms, being "main", are bigger or more important than the interactions, and again there's usually no reason to think that. Why we don't use names like "linear coefficients" and "product coefficients", I couldn't say.

Products without linear terms considered dubious It is very rare to find models where there is a product term $X_i X_j$ without both the linear terms X_i and X_j. If, say, the X_i term was missing, it would mean that Y was completely insensitive to X_i when $X_j = 0$, but only then. This is weird, and indeed flies in the face of one of the best justifications for using product interactions (§17.2.1). There's no intrinsic reason it couldn't happen, but you should expect models like that to receive additional scrutiny.

17.2.1 Why Product Interactions?

Most texts on linear regression do not even attempt to justify using interaction terms that look like $X_1 X_2$, as opposed to $\frac{X_1 X_2}{1+|X_1 X_2|}$, or $X_1 H(X_2 - c)$, etc., etc. Here is the best justification I can find.

H is the Heaviside step function, $H(x) = \left\{ \begin{array}{cc} 1 & x \geq 0 \\ 0 & x < 0 \end{array} \right.$.

Suppose that the real regression function $\mathbb{E}[Y|X = x] = \mu(x)$ is a smooth function of all the coordinates of x. Because it is smooth, we should be able to do a Taylor expansion around any particular point, say x^*:

$$\mu(x) \approx \mu(x^*) + \sum_{i=1}^{p}(x_i - x_i^*) \left.\frac{\partial \mu}{\partial x_i}\right|_{x=x^*} + \frac{1}{2}\sum_{i=1}^{p}\sum_{j=1}^{p}(x_i - x_i^*)(x_j - x_j^*) \left.\frac{\partial^2 \mu}{\partial x_i x_j}\right|_{x=x^*} \tag{17.4}$$

The first term, $\mu(x^*)$, is a constant. The next sum will give us linear terms in all the x_i (plus more constants). The double sum after that will give us terms for each product $x_i x_j$, plus all the squares x_i^2, plus more constants. Thus, if the true regression function is smooth, and we only see a small range of values for each predictor variable, using product terms is reasonable — provided we also include quadratic terms for each variable. (See Chapter 14 on polynomial regression for how to do that.)

Non-product interactions If have a particular sort of non-product interaction term in mind, say $\frac{X_1 X_2}{1+|X_1 X_2|}$, there is no particular difficulty in estimating it; just form a new column of predictors with the appropriate values, and estimate a coefficient on it like any other. Interpretation may, however, become even more tricky, and there is also the issue of deciding on what sort of interaction. Shalizi (forthcoming) describes ways of discovering reasonable interaction terms automatically, by two-dimensional smoothing.

17.3 Interaction of Categorical and Numerical Variables

If we multiply the indicator variable for a binary category, say X_B, with an ordinary numerical variable, say X_1, we get a different slope on X_i for each category:

$$Y = \beta_0 + \beta_1 X_1 + \beta_{1B} X_B X_1 + \epsilon \tag{17.5}$$

When $X_B = 0$, the slope on X_1 is β_1, but when $X_B = 1$, the slope on X_1 is $\beta_1 + \beta_{1B}$; the coefficient for the interaction is the *difference* in slopes between the two categories. This is just like the way the coefficients on categorical variables back in Chapter 14 ("adjusted effects") were differences between the intercepts for the categories.

In fact, look closely at Eq. 17.5. It says that the categories share a common *intercept*, but their regression lines are not parallel (unless $\beta_{1B} = 0$). We could expand the model by letting each category have its own slope and its own intercept:

$$Y = \beta_0 + \beta_B X_B + \beta_1 X_1 + \beta_{1B} X_B X_1 + \epsilon \tag{17.6}$$

This model, where "everything is interacted with the category", is *very* close to just running two separate regressions, one per category. It does, however, insist on having a single noise variance σ^2 (which separate regressions wouldn't accomplish). It also let you form confidence intervals for β_B and β_{1B}; if one or the other of these is tightly focused around 0, you might consider dropping that term and re-estimating[1]. Also,

[1]You *could* get the same effect with two separate regressions, by getting a confidence interval for the difference in the two estimates of the slope or the two estimates of the intercept, but the answer would come to the same as what you'd get from the joint regression with full interactions.

if there were additional predictors in the model which were not interacted with the category, e.g.,

$$Y = \beta_0 + \beta_B X_B + \beta_1 X_1 + \beta_{1B} X_B X_1 + \beta_2 X_2 + \epsilon \tag{17.7}$$

then this would definitely not be the same as running two separate regressions.

As with linear terms for categorical variables ("adjusted effects"), everything works much the same for variables with more than two levels: we add one indicator variable for all but one (reference or baseline) level of the category, we interact the indicators with the other predictor or predictors of interest, and the coefficients are differences to the slopes.

17.3.1 Interactions of Categorical Variables with Each Other

Nothing stops the variable you interact a categorical with from being another categorical. When that happens, you get terms which only apply to individuals which belong to *both* categories, e.g., to plumbers in Ohio.

Categorical interactions vs. group or conditional means Suppose we have two binary categorical variables, with corresponding indicator variables X_B and X_C. If we fit a model of the form

$$Y = \beta_0 + \beta_1 X_B + \beta_2 X_C + \beta_3 X_B X_C + \epsilon \tag{17.8}$$

then we can make the following identifications:

$$\begin{aligned}
\mathbb{E}[Y|X_B = 0, X_C = 0] &= \beta_0 && (17.9)\\
\mathbb{E}[Y|X_B = 1, X_C = 0] &= \beta_0 + \beta_1 && (17.10)\\
\mathbb{E}[Y|X_B = 0, X_C = 1] &= \beta_0 + \beta_2 && (17.11)\\
\mathbb{E}[Y|X_B = 1, X_C = 1] &= \beta_0 + \beta_1 + \beta_2 + \beta_3 && (17.12)
\end{aligned}$$

Conversely, these give us four equations in four unknowns, so if we know the group or conditional means on the left-hand sides, we could solve these equations for the βs (Exercise 2).

Notice that if our only predictor variables were these two categorical variables, we'd have one parameter for each distinct value of X — the model is **saturated** — and we'd have very little ability to tell that the model was wrong, regardless of how big n might be. One way we might check it would be to look at the distribution of residuals for each distinct group — by assumption they should all be the same. Of course if we have additional predictor variables, we can check the residuals against them.

17.4 Higher-Order Interactions

Nothing stops us from considering interactions among three or more variables, rather than just two. Again, the conventional form for this is a product, $X_i X_j X_k$. Again, the best justification for this I've ever seen is a higher-order Taylor expansion, which suggests using terms like $X_i^2 X_j$ and X_i^3 as well. Again, there is nothing special about diagnostics or inference for higher-order interaction terms. Trying to describe their interpretation in words gets extra tricky, however.

17.5 Product Interactions in R

The `lm` function is set up to comprehend multiplicative or product interactions in model formulas. Pure product interactions are denoted by `:`, so the formula

```
lm(y ~ x1:x2)
```

corresponds to the model $Y = \beta_0 + \beta X_1 X_2 + \epsilon$. (Intercepts are included by default in R.) Since it is relatively rare to include just a product term without linear terms, it's more common to use the symbol `*`, which expands out to both sets of terms. That is,

```
lm(y ~ x1 * x2)
```

is equivalent to

```
lm(y ~ x1 + x2 + x1:x2)
```

and both estimate the model $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \epsilon$. This special sense of `*` in formulas over-rides its ordinary sense of multiplication; if you wanted to specify a regression on, say $1000X_2$, you'd have to write `I(1000*x2)` rather than `1000*x2`. Also notice that R thinks, not unreasonably, that `x1:x1` is just the same as `x1`; if you want higher powers of a variable, use `I(x1^2)` or `poly(x1,2)`.

The `:` will apply to combinations of variables. Thus

```
(x1 + x2):(x3 + x4)
```

is equivalent to

```
x1:x3 + x1:x4 + x2:x3 + x2:x4
```

Similarly for `*`. This

```
(x1 + x2) * (x3 + x4)
```

expands out to this:

```
x1 + x2 + x3 + x4 + x1:x3 + x1:x4 + x2:x3 + x2:x4
```

The reason you can't just write `x1^2` in your model formula is that the power operator *also* has a special meaning in formulas, of repeatedly `*`-ing its argument with itself. That is, this

```
(x1 + x2 + x3)^2
```

is equivalent to

```
(x1 + x2 + x3) * (x1 + x2 + x3)
```

which in turn is equivalent to

```
x1 + x2 + x3 + x1:x2 + x1:x3 + x2:x3
```

(Remember that `x1:x1` is just `x1`.)

I find these operators in formulas most useful when I want to interact lots of variables with a category:

```
lm(y ~ (x1 + x2 + x3 + x5) * xcat + x4)
```

is a lot more compact than writing everything out, as

```
lm(y ~ xcat + x1 + x2 + x3 + x5 + x1:xcat + x2:xcat + x3:xcat + x5:xcat + x4)
```

and it's also something I'm a lot less likely to get wrong. Even writing out the whole formula term by term would be a lot less work, and lead to many fewer errors, than creating all the interacted columns by hand.

`poly` and interactions If you want to use `poly` to do polynomial regression, as in Chapter 14, *and* we want interactions, we can do it:

```
lm(y ~ poly(x1, x2, degree = 2))
```

This creates linear terms for both variables (which it gives names ending `1.0` and `0.1`), quadratic terms for both variables (names ending in `2.0` and `0.2`), and their product term (whose name ends in `1.1`). We have to explicitly name the `degree` argument; otherwise, `poly` doesn't know when we've stopped giving it columns we want to interact. If we set `degree` higher than 2, we'll get interactions between powers of the variables, and if we gave it $k > 2$ variables, we'd get all possible $2, 3, \ldots k$-way interactions.

17.5.1 Economic Mobility vs. Commuting, Again

Let's continue with the data from the first DAP.

```
mobility <- read.csv("http://www.stat.cmu.edu/~cshalizi/mreg/15/dap/1/mobility.csv")
```

As in Chapter 14, on categorical variables, we'll introduce a new binary category, indicating whether each state was or was not a part of the Confederacy in the Civil War. (See that chapter for detailed comments.)

```
# The states of the Confederacy
Confederacy <- c("AR", "AL", "FL", "GA", "LA", "MS", "NC", "SC", "TN", "TX",
    "VA")
mobility$Dixie <- mobility$State %in% Confederacy
```

In that chapter, we allowed this new indicator variable to change the intercept; you will recall that that term was negative and highly significant. Here, we'll let being in the South affect the slope on `Commute` as well, that is, we introduce an interaction between `Commute` and `Dixie`:

```
mob.dixie <- lm(Mobility ~ Commute * Dixie, data = mobility)
signif(coefficients(summary(mob.dixie)), 3)

##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        0.01880    0.00683  2.7600 5.95e-03
## Commute            0.19500    0.01340 14.5000 2.93e-42
## DixieTRUE         -0.02120    0.01190 -1.7700 7.64e-02
## Commute:DixieTRUE -0.00131    0.02830 -0.0461 9.63e-01
```

(See also Exercise 3.)

The coefficient for the interaction is negative, suggesting that increasing the fraction of workers with short commutes predicts a smaller difference in rates of mobility in the South than it does in the rest of the country. This coefficient is not significantly different from zero, but, more importantly, we can be confident it is small, compared to the base-line value of the slope on `Commute`:

```
signif(confint(mob.dixie), 3)

##                      2.5 %  97.5 %
## (Intercept)        0.00543 0.03220
## Commute            0.16900 0.22200
## DixieTRUE         -0.04470 0.00225
## Commute:DixieTRUE -0.05680 0.05420
```

Thus, even if the South does have a different slope than the rest of the country, it is not a very different slope.

The difference in the intercept, however, is more substantial. It, too, is not significant at the 5% level, but that is because (as we see from the confidence interval) it might be quite large and negative (—2 percentage points, when the mean is about 10% and the largest value is 47%), or perhaps just barely positive — it's not so precisely measured, but it's either lowering the expected rate of mobility or adding to it trivially.

Of course, we should really do all our diagnostics here before paying much attention to these inferential statistics, but I offer this by way of illustration of the functions. As a further illustraiton, see Exercise 4.

17.6 Exercises

1. Consider an apparently different, and perhaps more-interpretable, model than Eq. 17.3, namely

$$Y = \alpha_0 + (\alpha_1 + \alpha_2 X_2)X_1 + (\alpha_3 + \alpha_4 X_1)X_2 + \epsilon \tag{17.13}$$

   Show that this can always be re-written in the same form as Eq. 17.3, and express the latter's $\beta_0, \beta_1, \beta_2$ in terms of the αs of this model. Can models of the form of Eq. 17.3 always be re-written in this form? If so, express the α parameters in terms of the βs; if not, give a counter-example.

2. Solve Eqs. 17.9–17.12 for the βs.

3. Check that we get the same set of terms, with the same coefficients, as in §17.5.1, if we fit our model with

```
lm(Mobility ~ Commute + Dixie + Commute:Dixie, data = mobility)
```

   Why does this happen?

4. Using the mobility data, regress `Mobility` on

   (a) Latitude and longitude (only);

   (b) Latitude, longitude, and their product (only);

   (c) Latitude, longitude, their product, and their squares (only).

   For each model, make maps[2] of the fitted values and the residuals. Describe the resulting geographic patterns, and compare them (qualitatively) to a map of the actual values of `Mobility`. Can you explain why the maps of fitted values look like they do, based on the terms included in the model?

[2]See the hint on the DAP 1 assignment for help with making such maps.

Chapter 18

Outliers and Influential Points

An **outlier** is a data point which is very far, somehow, from the rest of the data. They are often worrisome, but not always a problem. When we are doing regression modeling, in fact, we don't really care about whether some data point is *far* from the rest of the data, but whether it *breaks a pattern* the rest of the data seems to follow. Here, we'll first try to build some intuition for when outliers cause trouble in linear regression models. Then we'll look at some ways of quantifying how much influence particular data points have on the model; consider the strategy of pretending that inconvenient data doesn't exist; and take a brief look at the **robust regression** strategy, of replacing least squares estimates with others which are less easily influenced.

18.1 Outliers Are Data Points Which Break a Pattern

Consider Figure 18.1. The points marked in red and blue are clearly not like the main cloud of the data points, even though their x and y coordinates are quite typical of the data as a whole: the x coordinates of those points aren't related to the y coordinates in the right way, they break a pattern. On the other hand, the point marked in green, while its coordinates are very weird on both axes, does not break that pattern — it was positioned to fall right on the regression line.

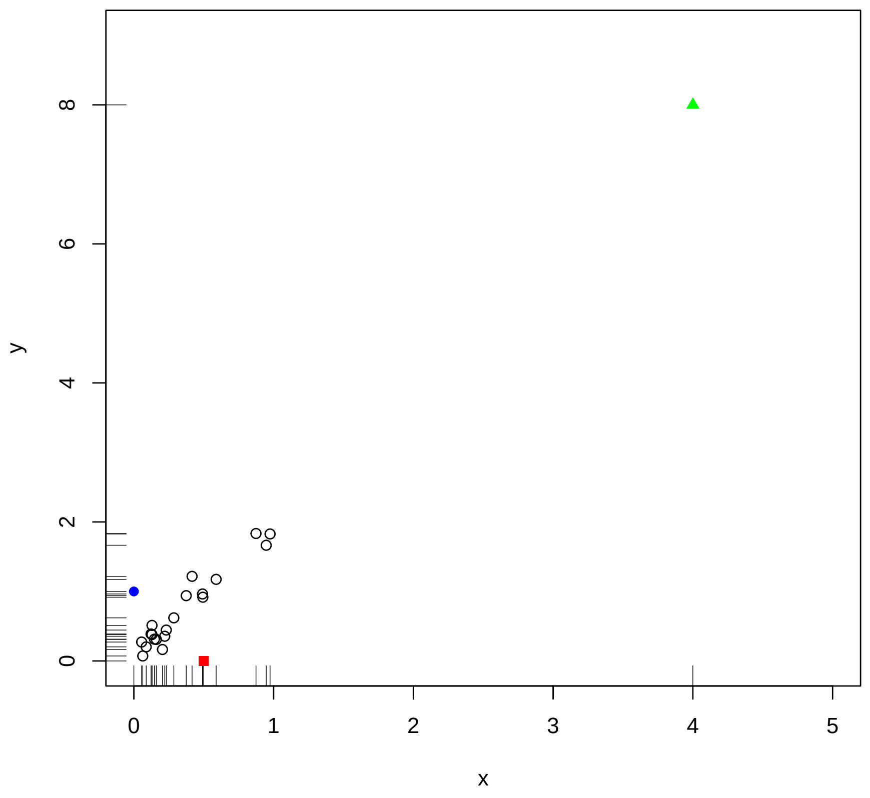


FIGURE 18.1: *Points marked with a red × and a blue triangle are outliers for the regression line through the main cloud of points, even though their x and y coordinates are quite typical of the marginal distributions (see rug plots along axes). The point marked by the green square, while an outlier along both axes, falls right along the regression line. (See the source file online for the figure-making code.)*

	(Intercept)	x
black only	0.0837	1.84
black+blue	0.0666	1.76
black+red	0.1850	1.67
black+green	0.0399	1.98
all points	0.0535	1.94

TABLE 18.1: *Estimates of the simple regression line from the black points in Figure 18.1, plus re-estimates adding in various outliers.*

What affect do these different outliers have on a simple linear model here? Table 18.1 shows the estimates we get from using just the black points, from adding only one of the three outlying points to the black points, and from using all the points. As promised, adding the red or blue points shifts the line, while adding the green point changes hardly anything at all.

If we are worried that outliers might be messing up our model, we would like to quantify how much the estimates change if we add or remove individual data points. Fortunately, we can quantify this using only quantities we estimated on the complete data, especially the hat matrix.

18.1.1 Examples with Simple Linear Regression

To further build intuition, let's think about what happens with simple linear regression for a moment; that is, our model is

$$Y = \beta_0 + \beta_1 X + \epsilon \tag{18.1}$$

with a single real-valued predictor variable X. When we estimate the coefficients by least squares, we know that

$$\hat{\beta}_0 = \overline{y} - \hat{\beta}_1 \overline{x} \tag{18.2}$$

Let us turn this around. The fitted value at $X = \overline{x}$ is

$$\hat{\beta}_0 + \hat{\beta}_1 \overline{x} = \overline{y} \tag{18.3}$$

Suppose we had a data point, say the i^{th} point, where $X = \overline{x}$. Then the actual value of y_i almost wouldn't matter for the fitted value there — the regression line *has* to go through $\overline{y}$ at $\overline{x}$, never mind whether y_i there is close to $\overline{y}$ or far away. If $x_i = \overline{x}$, we say that y_i has little **leverage** over $\hat{m}_i$, or little **influence** on $\hat{m}_i$. It has *some* influence, because y_i is part of what we average to get $\overline{y}$, but that's not a lot of influence.

Again, with simple linear regression, we know that

$$\hat{\beta}_1 = \frac{c_{XY}}{s^2_X} \tag{18.4}$$

the ratio between the sample covariance of X and Y and the sample variance of X. How does y_i show up in this? It's

$$\hat{\beta}_1 = \frac{n^{-1}\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{s^2_X} \tag{18.5}$$

Notice that when $x_i = \overline{x}$, y_i doesn't actually matter at all to the slope. If x_i is far from $\overline{x}$, then $y_i - \overline{y}$ will contribute to the slope, and its contribution will get bigger (whether positive or negative) as $x_i - \overline{x}$ grows. y_i will also make a big contribution to the slope when $y_i - \overline{y}$ is big (unless, again, $x_i = \overline{x}$).

Let's write a general formula for the predicted value, at an arbitrary point $X = x$.

$$\begin{aligned}
\hat{m}(x) &= \hat{\beta}_0 + \hat{\beta}_1 x && (18.6)\\
&= \overline{y} - \hat{\beta}_1 \overline{x} + \hat{\beta}_1 x && (18.7)\\
&= \overline{y} + \hat{\beta}_1 (x - \overline{x}) && (18.8)\\
&= \overline{y} + \frac{1}{n}\frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{s^2_X}(x - \overline{x}) && (18.9)
\end{aligned}$$

So, in words:

- The predicted value is always a weighted average of all the y_i.

- As x_i moves away from $\overline{x}$, y_i gets more weight (possibly a large negative weight). When $x_i = \overline{x}$, y_i only matters because it contributes to the global mean $\overline{y}$.
- The weights on all data points increase in magnitude when the point x where we're trying to predict is far from $\overline{x}$. If $x = \overline{x}$, only $\overline{y}$ matters.

All of this is still true of the fitted values at the original data points:

- If x_i is at $\overline{x}$, y_i only matters for the fit because it contributes to $\overline{y}$.
- As x_i moves away from $\overline{x}$, in either direction, it makes a bigger contribution to *all* the fitted values.

Why is this happening? We get the coefficient estimates by minimizing the mean squared error, and the MSE treats all data points equally:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{m}(x_i))^2 \tag{18.10}$$

But we're not just using any old function $\hat{m}(x)$; we're using a linear function. This has only two parameters, so we can't change the predicted value to match each data point — altering the parameters to bring $\hat{m}(x_i)$ closer to y_i might actually increase the error elsewhere. By minimizing the over-all MSE with a linear function, we get two constraints,

$$\overline{y} = \hat{\beta}_0 + \hat{\beta}_1\overline{x} \tag{18.11}$$

and

$$\sum_i e_i(x_i - \overline{x}) = 0 \tag{18.12}$$

The first of these makes the regression line insensitive to y_i values when x_i is close to $\overline{x}$. The second makes the regression line *very* sensitive to residuals when $x_i - \overline{x}$ is big — when $x_i - \overline{x}$ is large, a big residual (e_i far from 0) is harder to balance out than if $x_i - \overline{x}$ were smaller.

So, let's sum this up.

- Least squares estimation tries to bring all the predicted values closer to y_i, but it can't match each data point at once, because the fitted values are all functions of the same coefficients.
- If x_i is close to $\overline{x}$, y_i makes little difference to the coefficients or fitted values — they're pinned down by needing to go through the mean of the data.
- As x_i moves away from $\overline{x}$, $y_i - \overline{y}$ makes a bigger and bigger impact on both the coefficients and on the fitted values.

If we worry that some point isn't falling on the same regression line as the others, we're really worrying that including it will throw off our estimate of the line. This is going to be a concern when x_i is far from $\overline{x}$, or when the combination of $x_i - \overline{x}$ and $y_i - \overline{y}$ makes that point have a disproportionate impact on the estimates. We should also be worried if the residual values are too big, but when asking what's "too big", we need to take into account the fact that the model will try harder to fit some points than others. A big residual at a point of high leverage is more of a red flag than an equal-sized residual at point with little influence.

All of this will carry over to multiple regression models, but with more algebra to keep track of the different dimensions.

18.2 Influence of Individual Data Points on Estimates

Recall that our least-squares coefficient estimator is

$$\widehat{\beta} = (\mathbf{x}^T \mathbf{x})^{-1} \mathbf{x}^T \mathbf{y} \tag{18.13}$$

from which we get our fitted values as

$$\widehat{\mathbf{m}} = \mathbf{x}\widehat{\beta} = \mathbf{x}(\mathbf{x}^T \mathbf{x})^{-1} \mathbf{x}^T \mathbf{y} = \mathbf{H}\mathbf{y} \tag{18.14}$$

with the hat matrix $\mathbf{H} \equiv \mathbf{x}(\mathbf{x}^T \mathbf{x})^{-1} \mathbf{x}^T$. This leads to a very natural sense in which one observation might be more or less influential than another:

$$\frac{\partial \hat{\beta}_k}{\partial y_i} = \left((\mathbf{x}^T \mathbf{x})^{-1} \mathbf{x}^T\right)_{ki} \tag{18.15}$$

and

$$\frac{\partial \hat{m}_k}{\partial y_i} = H_{ki} \tag{18.16}$$

If y_i were different, it would change the estimates for all the coefficients and for all the fitted values. The rate at which the k^{th} coefficient or fitted value changes is given by the ki^{th} entry in these matrices — matrices which, notice, are completely defined by the design matrix $\mathbf{x}$.

18.2.1 Leverage

H_{ii} is the influence of y_i on its own fitted value; it tells us how much of $\hat{m}_i$ is just y_i. This turns out to be a key quantity in looking for outliers, so we'll give it a special name, the **leverage**. It is sometimes also written h_i. Once again, the leverage of the i^{th} data point doesn't depend on y_i, only on the design matrix.

Because the general linear regression model doesn't assume anything about the distribution of the predictors, other than that they're not collinear, we can't say definitely that some values of the leverage break model assumptions, or even are very unlikely under the model assumptions. But we can say some things about the leverage.

Average leverages Exercise 6 in Chapter 12 showed that the trace of the hat matrix equals the number of coefficients we estimate:

$$\operatorname{tr}\mathbf{H} = p+1 \tag{18.17}$$

But the trace of any matrix is the sum of its diagonal entries,

$$\operatorname{tr}\mathbf{H} = \sum_{i=1}^{n} H_{ii} \tag{18.18}$$

so the trace of the hat matrix is the sum of each point's leverage. The average leverage is therefore $\frac{p+1}{n}$. We don't expect every point to have exactly the same leverage, but if some points have much more than others, the regression function is going to be pulled towards fitting the high-leverage points, and the function will tend to ignore the low-leverage points.

Leverage vs. geometry Let's center all the predictor variables, i.e., subtract off the mean of each predictor variable. Call this new vector of predictor variables Z, with the $n \times p$ matrix $\mathbf{z}$. This will not change any of the slopes, and will fix the intercept to be $\overline{y}$. The fitted values then come from

$$\hat{m}_i = \overline{y} + \frac{1}{n}(\mathbf{x}_i - \overline{\mathbf{x}})\operatorname{Var}[X]^{-1}\mathbf{z}^T\mathbf{y} \tag{18.19}$$

This tells us that y_i will have a lot of leverage if $(\mathbf{x}_i - \overline{\mathbf{x}})\operatorname{Var}[X]^{-1}(\mathbf{x}_i - \overline{\mathbf{x}})^T$ is big[1]. If the data point falls exactly at the mean of the predictors, y_i matters only because it contributes to the over-all mean $\overline{y}$. If the data point moves away from the mean of the predictors, not all directions count equally. Remember the eigen-decomposition of $\operatorname{Var}[X]$:

$$\operatorname{Var}[X] = \mathbf{V}\mathbf{U}\mathbf{V}^T \tag{18.20}$$

where $\mathbf{V}$ is the matrix whose columns are the eigenvectors of $\operatorname{Var}[X]$, $\mathbf{V}^T = \mathbf{V}^{-1}$, and $\mathbf{U}$ is the diagonal matrix of the eigenvalues of $\operatorname{Var}[X]$. Each eigenvalue gives the variance of the predictors along the direction of the corresponding eigenvector. It follows that

$$\operatorname{Var}[X]^{-1} = \mathbf{V}\mathbf{U}^{-1}\mathbf{V} \tag{18.21}$$

So if the data point is far from the center of the predictors along a high-variance direction, that doesn't count as much as being equally far along a low-variance direction[2]. Figure 18.2 shows a distribution for two predictor variables we're very familiar with, together with the two eigenvectors from the variance matrix, and the corresponding surface of leverages.

You may convince yourself that with one predictor variable, all of this collapses down to just $1/n + (x_i - \overline{x})^2/ns^2_X$ (Exercise 1). This leads to plots which may be easier to grasp (Figure 18.3).

[1]This sort of thing — take the difference between two vectors, multiply by an inverse variance matrix, and multiply by the difference vector again — is called a **Mahalanobis distance**. As we will see in a moment, it gives more attention to differences along coordinates where the variance is small, and less attention to differences along coordinates where the variance is high.

[2]I have an unfortunate feeling that I said this backwards throughout the afternoon.

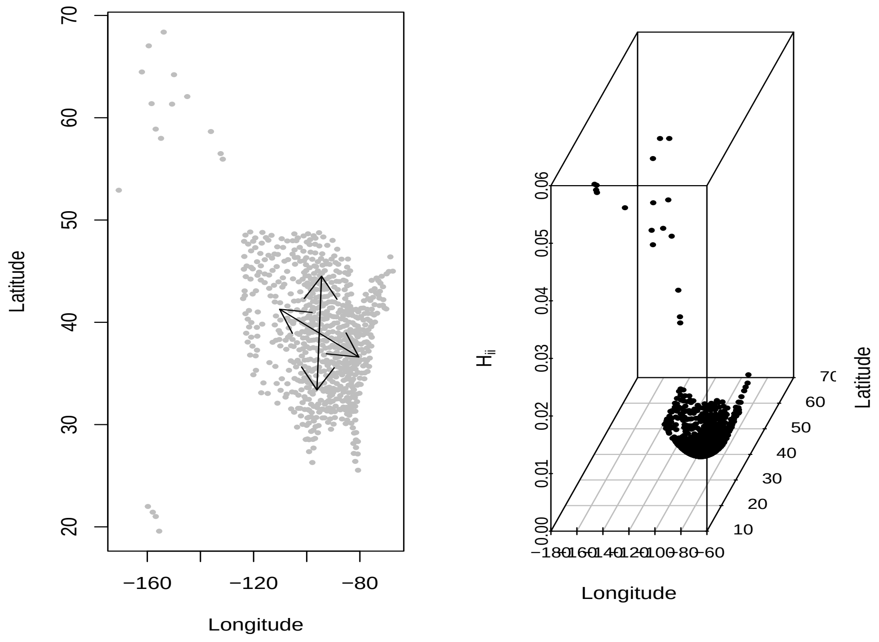
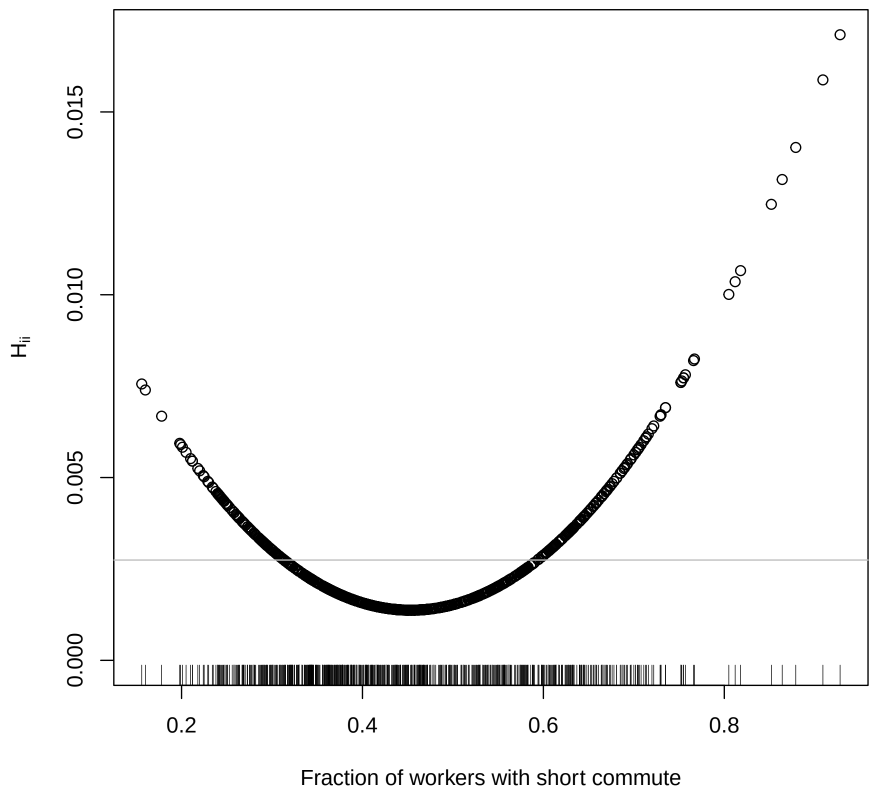


FIGURE 18.2: *Left: The geographic coordinates of the communities from the* `mobility` *data, along with their mean, and arrows marking the eigenvectors of the variance-covariance matrix (lengths scaled by the eigenvalues). Right: leverages for each point when regressing rates of economic mobility (or anything else) on latitude and longitude. See online for the code.*



```
H.mob.lm <- hatvalues(lm(Mobility ~ Commute, data = mobility))
plot(mobility$Commute, H.mob.lm, ylim = c(0, max(H.mob.lm)), xlab = "Fraction of workers with short
    ylab = expression(H[ii]))
abline(h = 2/nrow(mobility), col = "grey")
rug(mobility$Commute, side = 1)
```

FIGURE 18.3: *Leverages (H_{ii}) for a simple regression of economic mobility (or anything else) against the fraction of workers with short commutes. The grey line marks the average we'd see if every point was exactly equally influential. Note how leverage increases automatically as* `Commute` *moves away from its mean in either direction. (See below for the* `hatvalues` *function.*

One curious feature of the leverage is, and of the hat matrix in general, is that it doesn't care *what* we are regressing on the predictor variables; it could be economic mobility or sightings of Bigfoot, and the same design matrix will give us the same hat matrix and leverages.

To sum up: The leverage of a data point just depends on the value of the predictors there; it increases as the point moves away from the mean of the predictors. It increases more if the difference is along low-variance coordinates, and less for differences along high-variance coordinates.

18.3 Studentized Residuals

We return once more to the hat matrix, the source of all knowledge.

$$\widehat{\mathbf{m}} = \mathbf{H}\mathbf{y} \tag{18.22}$$

The residuals, too, depend only on the hat matrix:

$$\mathbf{e} = \mathbf{y} - \widehat{\mathbf{m}} = (\mathbf{I} - \mathbf{H})\mathbf{y} \tag{18.23}$$

We know that the residuals vary randomly with the noise, so let's re-write this in terms of the noise (Exercise 2).

$$\mathbf{e} = (\mathbf{I} - \mathbf{H})\epsilon \tag{18.24}$$

Since $\mathbb{E}[\epsilon] = 0$ and $\mathrm{Var}[\epsilon] = \sigma^2\mathbf{I}$, we have

$$\mathbb{E}[\mathbf{e}] = 0 \tag{18.25}$$

and

$$\mathrm{Var}[\mathbf{e}] = \sigma^2(\mathbf{I} - \mathbf{H})(\mathbf{I} - \mathbf{H})^T = \sigma^2(\mathbf{I} - \mathbf{H}) \tag{18.26}$$

If we also assume that the noise is Gaussian, the residuals are Gaussian, with the stated mean and variance.

What does this imply for the residual at the i^{th} data point? It has expectation 0,

$$\mathbb{E}[e_i] = 0 \tag{18.27}$$

and it has a variance which depends on i through the hat matrix:

$$\mathrm{Var}[e_i] = \sigma^2(\mathbf{I} - \mathbf{H})_{ii} = \sigma^2(1 - H_{ii}) \tag{18.28}$$

In words: the bigger the leverage of i, the smaller the variance of the residual there. This is yet another sense in which points with high leverage are points which the model tries very hard to fit.

Previously, when we looked at the residuals, we expected them to all be of roughly the same magnitude. This rests on the leverages H_{ii} being all about the same size. If there are substantial variations in leverage across the data points, it's better to scale the residuals by their expected size.

The usual way to do this is through the **standardized** or **studentized residuals**

$$r_i \equiv \frac{e_i}{\hat{\sigma}\sqrt{1-H_{ii}}} \tag{18.29}$$

Why "studentized"? Because we're dividing by an estimate of the standard error, just like in "Student's" t-test for differences in means[3]

All of the residual plots we've done before can also be done with the studentized residuals. In particular, the studentized residuals should look flat, with constant variance, when plotted against the fitted values or the predictors.

18.4 Leave-One-Out

Suppose we left out the i^{th} data point altogether. How much would that change the model?

18.4.1 Fitted Values and Cross-Validated Residuals

Let's take the fitted values first. The hat matrix, $\mathbf{H}$, is an $n \times n$ matrix. If we deleted the i^{th} observation when estimating the model, but still asked for a prediction at $\mathbf{x}_i$, we'd get a different, $n \times (n-1)$ matrix, say $\mathbf{H}^{(-i)}$. This in turn would lead to a new fitted value:

$$\hat{m}^{(-i)}(\mathbf{x}_i) = \frac{(\mathbf{Hy})_i - \mathbf{H}_{ii}y_i}{1-\mathbf{H}_{ii}} \tag{18.30}$$

Basically, this is saying we can take the old fitted value, and then subtract off the part of it which came from having included the observation y_j in the first place. Because each row of the hat matrix has to add up to 1 (Exercise 3), we need to include the denominator (Exercise 4).

The **leave-one-out residual** is the difference between this and y_i:

$$e_i^{(-i)} \equiv y_i - \hat{m}^{(-i)}(\mathbf{x}_i) \tag{18.31}$$

That is, this is how far off the model's prediction of y_i would be if it didn't actually get to see y_i during the estimation, but had to honestly predict it.

Leaving out the data point i would give us an MSE of $\hat{\sigma}^2_{(-i)}$, and a little work says that

$$t_i \equiv \frac{e_i^{(-i)}}{\hat{\sigma}_{(-i)}\sqrt{1+\mathbf{x}_i^T(\mathbf{x}_{(-i)}^T\mathbf{x}_{(-i)})^{-1}\mathbf{x}_i}} \; t_{n-p-2} \tag{18.32}$$

[3]The distribution here is however not quite a t-distribution, because, while e_i has a Gaussian distribution and $\hat{\sigma}$ is the square root of a χ^2-distributed variable, e_i is actually used in computing $\hat{\sigma}$, hence they're not statistically independent. Rather, $r_i^2/(n-p-1)$ has a $\beta(\frac{1}{2}, \frac{1}{2}(n-p-2))$ distribution (Seber and Lee, 2003, p. 267). This gives us studentized residuals which all have the same distribution, and that distribution does approach a Gaussian as $n \to \infty$ with p fixed.

(The -2 here is because these predictions are based on only $n-1$ data points.) These are called the **cross-validated**, or **jackknife**, or **externally studentized**, residuals. (Some people use the name "studentized residuals" only for these, calling the others the "standardized residuals".) Fortunately, we can compute this without having to actually re-run the regression:

$$t_i = \frac{e_i^{(-i)}}{\hat{\sigma}_{(-i)}\sqrt{1+\mathbf{x}_i^T(\mathbf{x}_{(-i)}^T\mathbf{x}_{(-i)})^{-1}\mathbf{x}_i}} \quad (18.33)$$

$$= \frac{e_i}{\hat{\sigma}_{(-i)}\sqrt{1-H_{ii}}} \quad (18.34)$$

$$= r_i\sqrt{\frac{n-p-2}{n-p-1-r_i^2}} \quad (18.35)$$

18.4.2 Cook's Distance

Omitting point i will generally change all of the fitted values, not just the fitted value at that point. We go from the vector of predictions $\widehat{\mathbf{m}}$ to $\widehat{\mathbf{m}}^{(-i)}$. How big a change is this? It's natural (by this point!) to use the squared length of the difference vector,

$$\|\widehat{\mathbf{m}}-\widehat{\mathbf{m}}^{(-i)}\|^2 = (\widehat{\mathbf{m}}-\widehat{\mathbf{m}}^{(-i)})^T(\widehat{\mathbf{m}}-\widehat{\mathbf{m}}^{(-i)}) \quad (18.36)$$

To make this more comparable across data sets, it's conventional to divide this by $(p+1)\hat{\sigma}^2$, since there are really only $p+1$ independent coordinates here, each of which might contribute something on the order of $\hat{\sigma}^2$. This is called the **Cook's distance** or **Cook's statistic** for point i:

$$D_i = \frac{(\widehat{\mathbf{m}}-\widehat{\mathbf{m}}^{(-i)})^T(\widehat{\mathbf{m}}-\widehat{\mathbf{m}}^{(-i)})}{(p+1)\hat{\sigma}^2} \quad (18.37)$$

As usual, there is a simplified formula, which evades having to re-fit the regression:

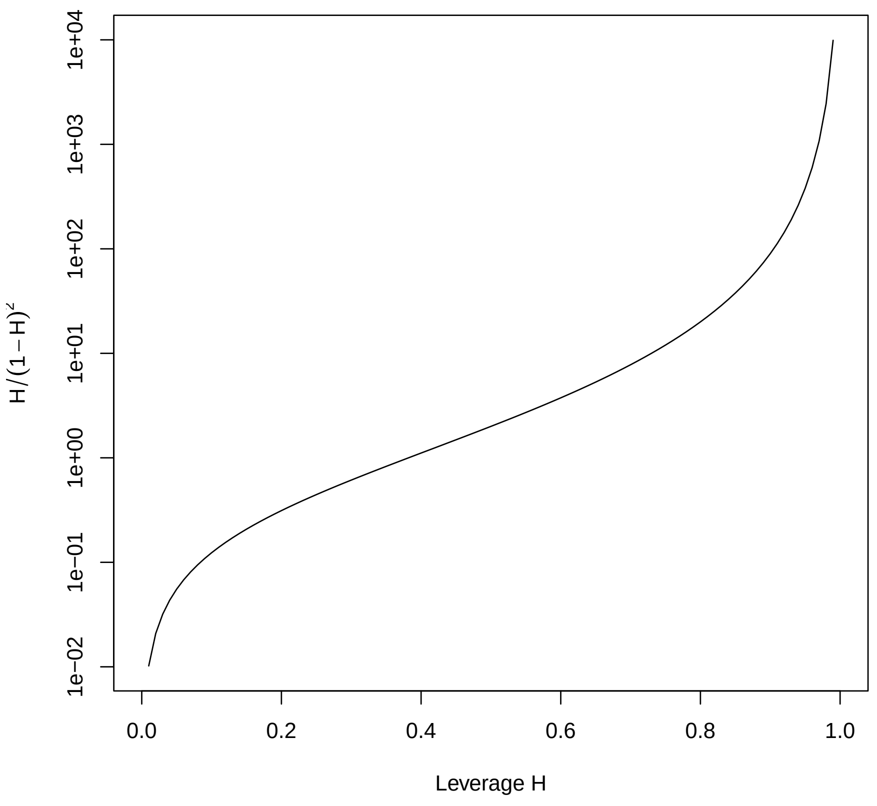
$$D_i = \frac{1}{p+1}e_i^2\frac{H_{ii}}{(1-H_{ii})^2} \quad (18.38)$$

Notice that $H_{ii}/(1-H_{ii})^2$ is a growing function of H_{ii} (Figure 18.4). So this says that the total influence of a point over all the fitted values grows with both its leverage (H_{ii}) and the size of its residual when it is included (e_i^2).

18.4.3 Coefficients

The leave-one-out idea can also be applied to the coefficients. Writing $\widehat{\beta}^{(-i)}$ for the vector of coefficients we get when we drop the ith data point. One can show (Seber and Lee, 2003, p. 268) that

$$\widehat{\beta}^{(-i)} = \widehat{\beta} - \frac{(\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}_i^T e_i}{1-H_{ii}} \quad (18.39)$$



```
curve(x/(1 - x)^2, from = 0, to = 1, xlab = "Leverage H", ylab = expression(H/(1 -
    H)^2), log = "y")
```

FIGURE 18.4: *Illustration of the function $H/(1-H)^2$ relating leverage H to Cook's distance. Notice that leverage must be ≥ 0 and ≤ 1, so this is the whole relevant range of the curve.*

Cook's distance can actually be computed from this, since the change in the vector of fitted values is $\mathbf{x}(\widehat{\beta}^{(-i)} - \widehat{\beta})$, so

$$D_i = \frac{((\widehat{\beta}^{(-i)} - \widehat{\beta})^T \mathbf{x}^T \mathbf{x}(\widehat{\beta}^{(-i)} - \widehat{\beta})}{(p+1)\hat{\sigma}^2} \tag{18.40}$$

18.4.4 Leave-More-Than-One-Out

Sometimes, whole clusters of nearby points might be potential outliers. In such cases, removing just one of them might change the model very little, while removing them all might change it a great deal. Unfortunately there are $\binom{n}{k} = O(n^k)$ groups of k points you could consider deleting at once, so while looking at all leave-one-out results is feasible, looking at all leave-two- or leave-ten- out results is not. Instead, you have to think.

18.5 Practically, and with R

We have three ways of looking at whether points are outliers:

1. We can look at their leverage, which depends only on the value of the predictors.
2. We can look at their studentized residuals, either ordinary or cross-validated, which depend on how far they are from the regression line.
3. We can look at their Cook's statistics, which say how much removing each point shifts all the fitted values; it depends on the product of leverage and residuals.

The model assumptions don't put any limit on how big the leverage can get (just that it's ≤ 1 at each point) or on how its distributed across the points (just that it's got to add up to $p+1$). Having most of the leverage in a few super-inferential points doesn't break the model, exactly, but it should make us worry.

The model assumptions *do* say how the studentized residuals should be distributed. In particular, the cross-validated studentized residuals should follow a t distribution. This is something we can test, either for specific points which we're worried about (say because they showed up on our diagnostic plots), or across all the points[4].

Because Cook's distance is related to how much the parameters change, the theory of confidence ellipsoids (Chapter 16) can be used to get some idea of how big a D_i is worrying[5]. Cook's original rule-of-thumb translates into worrying when $(p+1)D_i$ is

[4]Be careful about testing all the points. If you use a size α test and everything is fine, you'd see about αn rejections. A good, if not necessarily optimal, way to deal with this is to lower the threshold to α/n for each test — another example of the Bonferroni correction from Chapter 16.

[5]Remember we saw that for large n, $(\widehat{\beta} - \beta)^T \mathbf{\Sigma}^{-1}(\widehat{\beta} - \beta) \sim \chi^2_{p+1}$, where $\mathbf{\Sigma}$ is the variance matrix of the coefficient estimates. But that's $\sigma^2(\mathbf{x}^T\mathbf{x})^{-1}$, so we get $\sigma^{-2}(\widehat{\beta} - \beta)^T \mathbf{x}^T \mathbf{x}(\widehat{\beta} - \beta) \sim \chi^2_{p+1}$. Now compare with Eq. 18.40.

bigger than about $\chi^2_{p+1}(0.1)$, though the 0.1 is arbitrary[6]. However, this is not really a hypothesis test.

18.5.1 In R

Almost everything we've talked — leverages, studentized residuals, Cook's statistics — can be calculated using the `influence` function. However, there are more user-friendly functions which call that in turn, and are probably better to use.

Leverages come from the 'hatvalues' function, or from the 'hat' component of what 'influence' returns:

```
mob.lm <- lm(Mobility ~ Commute, data = mobility)
hatvalues(mob.lm)
influence(mob.lm)$hat  # Same as previous line
```

The standardized, or internally-studentized, residuals r_i are available with `rstandard`:

```
rstandard(mob.lm)
residuals(mob.lm)/sqrt(1 - hatvalues(mob.lm))  # Same as previous line
```

The cross-validated or externally-studentized residuals t_i are available with `rstudent`:

```
rstudent(mob.lm)  # Too tedious to calculate from rstandard though you could
```

Cook's statistic is calculated with `cooks.distance`:

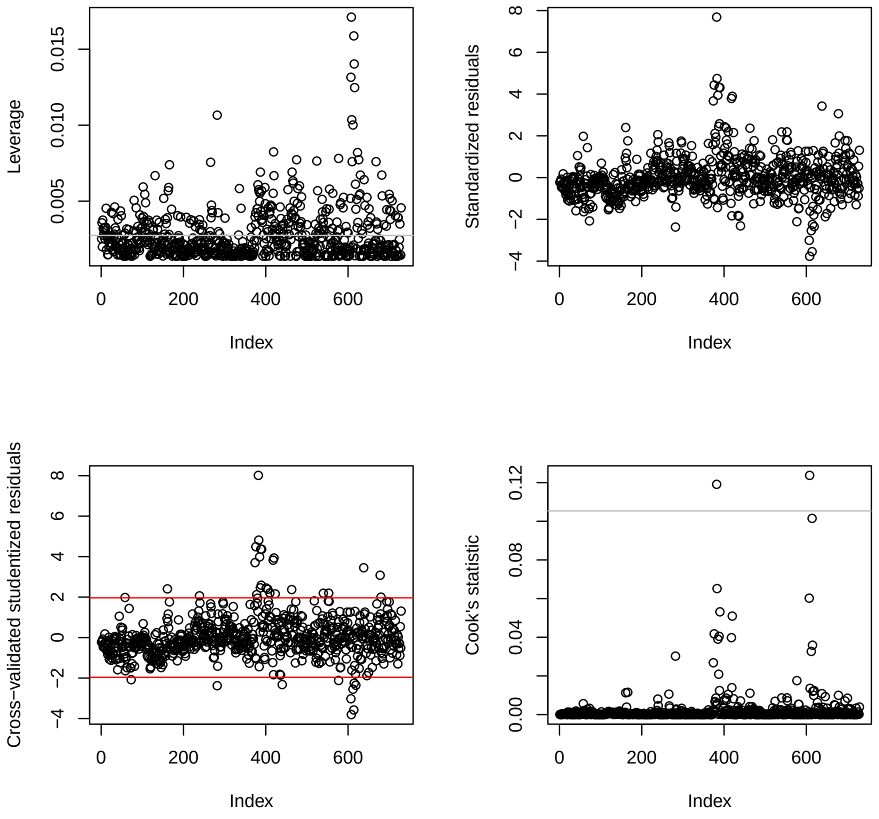
```
cooks.distance(mob.lm)
```

Often the most useful thing to do with these is to plot them, and look at the most extreme points. (One might also rank them, and plot them against ranks.) Figure 18.5 does so. The standardized and studentized residuals can also be put into our usual diagnostic plots, since they should average to zero and have constant variance when plotted against the fitted values or the predictors. (I omit that here because in this case, $1/\sqrt{1-H_{ii}}$ is sufficiently close to 1 that it makes no visual difference.)

We can now look at exactly which points have the extreme values, say the 10 most extreme residuals, or largest Cook's statistics:

```
mobility[rank(-abs(rstudent(mob.lm)), ) <= 10, ]
##       X       Name   Mobility State Commute  Longitude Latitude
## 374 375     Linton 0.29891303    ND   0.646 -100.16075 46.31258
## 376 378 Carrington 0.33333334    ND   0.656  -98.86684 47.59698
## 382 384     Bowman 0.46969697    ND   0.648 -103.42526 46.33993
## 383 385     Lemmon 0.35714287    ND   0.704 -102.42011 45.96558
```

[6]More exactly, he used an F distribution to take account of small-n uncertainties in $\hat{\sigma}^2$, and suggested worrying when D_i was bigger than $F_{p+1,n-p-1}(0.1)$. This will come to the same thing for large n.



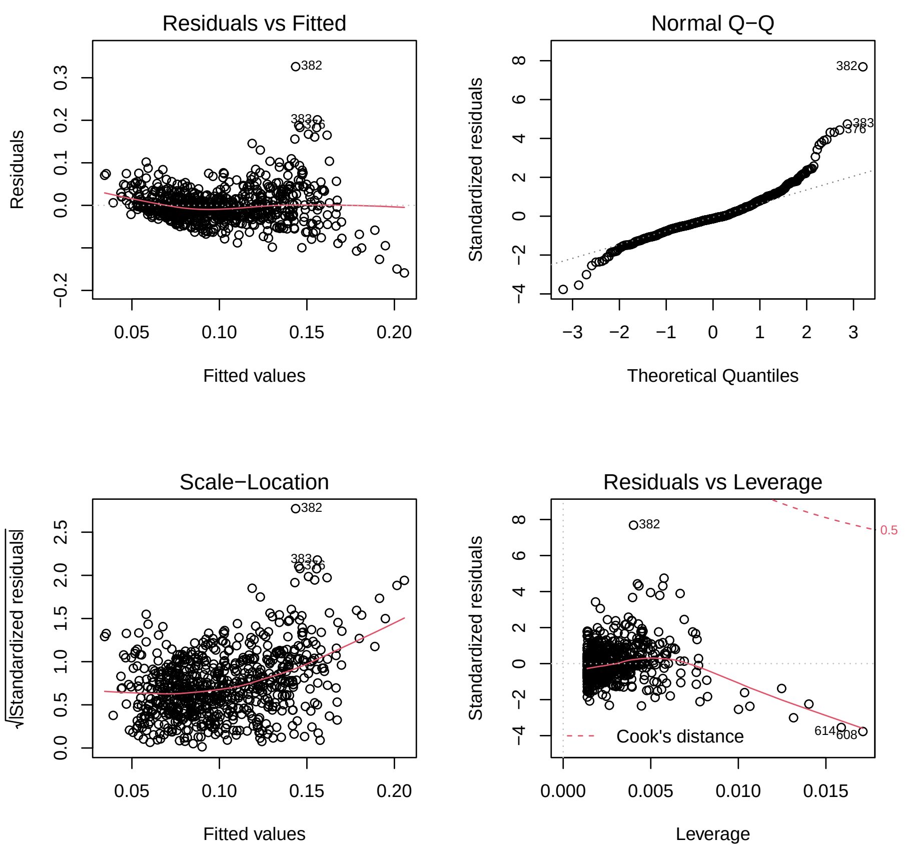
```
par(mfrow = c(2, 2))
mob.lm <- lm(Mobility ~ Commute, data = mobility)
plot(hatvalues(mob.lm), ylab = "Leverage")
abline(h = 2/nrow(mobility), col = "grey")
plot(rstandard(mob.lm), ylab = "Standardized residuals")
plot(rstudent(mob.lm), ylab = "Cross-validated studentized residuals")
abline(h = qt(0.025, df = nrow(mobility) - 2), col = "red")
abline(h = qt(1 - 0.025, df = nrow(mobility) - 2), col = "red")
plot(cooks.distance(mob.lm), ylab = "Cook's statistic")
abline(h = qchisq(0.1, 2)/2, col = "grey")
```

FIGURE 18.5: *Leverages, two sorts of standardized residuals, and Cook's distance statistic for each point in a basic linear model of economic mobility as a function of the fraction of workers with short commutes. The horizontal line in the plot of leverages shows the average leverage. The lines in studentized residual plot shows a 95% t-distribution sampling interval. (What is the grey line in the plot of Cook's distances?) Note the clustering of extreme residuals* and *leverage around row 600, and another cluster of points with extreme residuals around row 400.*

```
## 385 388 Plentywood 0.31818181    MT   0.681 -104.65381 48.64743
## 388 391  Dickinson 0.32920793    ND   0.659 -102.61354 47.32696
## 390 393  Williston 0.33830845    ND   0.702 -103.33987 48.25441
## 418 422     Miller 0.31506848    SD   0.697  -99.27758 44.53313
## 420 424 Gettysburg 0.32653061    SD   0.729 -100.19547 45.05100
## 608 618       Nome 0.04678363    AK   0.928 -162.03012 64.47514
mobility[rank(-abs(cooks.distance(mob.lm))) <= 10, ]
##       X       Name   Mobility State Commute  Longitude Latitude
## 376 378 Carrington 0.33333334    ND   0.656  -98.86684 47.59698
## 382 384     Bowman 0.46969697    ND   0.648 -103.42526 46.33993
## 383 385     Lemmon 0.35714287    ND   0.704 -102.42011 45.96558
## 388 391  Dickinson 0.32920793    ND   0.659 -102.61354 47.32696
## 390 393  Williston 0.33830845    ND   0.702 -103.33987 48.25441
## 418 422     Miller 0.31506848    SD   0.697  -99.27758 44.53313
## 420 424 Gettysburg 0.32653061    SD   0.729 -100.19547 45.05100
## 607 617   Kotzebue 0.06451613    AK   0.864 -159.43781 67.02818
## 608 618       Nome 0.04678363    AK   0.928 -162.03012 64.47514
## 614 624     Bethel 0.05186386    AK   0.909 -158.38213 61.37712
```

18.5.2 `plot`

We have not used the `plot` function on an `lm` object yet. This is because most of what it gives us is in fact related to residuals (Figure 18.6). The first plot is of residuals versus fitted values, plus a smoothing line, with extreme residuals marked by row number. The second is a Q-Q plot of the standardized residuals, again with extremes marked by row number. The third shows the square root of the absolute standardized residuals against fitted values (ideally, flat); the fourth plots standardized residuals against leverage, with contour lines showing equal values of Cook's distance. There are many options, described in `help(plot.lm)`.



```
par(mfrow = c(2, 2))
plot(mob.lm)
par(mfrow = c(1, 1))
```

FIGURE 18.6: *The basic* `plot` *function applied to our running example model.*

18.6 Responses to Outliers

There are essentially three things to do when we're convinced there are outliers: delete them; change the model; or change how we estimate.

18.6.1 Deletion

Deleting data points should never be done lightly, but it is sometimes the right thing to do.

The best case for removing a data point is when you have good reasons to think it's just wrong (and you have no way to fix it). Medical records which give a patient's blood pressure as 0, or their temperature as 200 degrees, are just impossible and have to be errors[7]. Those points aren't giving you useful information about the process you're studying[8], so getting rid of them makes sense.

The next best case is if you have good reasons to think that the data point isn't *wrong*, exactly, but belongs to a different phenomenon or population from the one you're studying. (You're trying to see if a new drug helps cancer patients, but you discover the hospital has included some burn patients and influenza cases as well.) Or the data point does belong to the right population, but also somehow to another one which isn't what you're interested in right now. (All of the data is on cancer patients, but some of them were also sick with the flu.) You should be careful about that last, though. (After all, some proportion of future cancer patients are also going to have the flu.)

The next best scenario after that is that there's nothing quite so definitely wrong about the data point, but it just looks really weird compared to all the others. Here you are really making a judgment call that either the data really are mistaken, or not from the right population, but you can't put your finger on a concrete reason why. The rules-of-thumb used to identify outliers, like "Cook's distance shouldn't be too big", or "Tukey's rule"[9], are at best of this sort. It is always more satisfying, and more reliable, if investigating how the data were gathered lets you turn cases of this sort into one of the two previous kinds.

The least good case for getting rid of data points which isn't just bogus is that you've got a model which almost works, and would work a lot better if you just get rid of a few stubborn points. This is really a sub-case of the previous one, with added special pleading on behalf of your favorite model. You are here basically trusting your model more than your data, so it had better be either a really good model or really bad data.

Beyond this, we get into what can only be called ignoring inconvenient facts so that you get the answer you want.

[7]This is true whether the temperature is in degrees Fahrenheit, degrees centigrade, or kelvins.

[8]Unless it's the very process of making errors of measurement and recording.

[9]Which flags any point more than 1.5 times the inter-quartile range above the third quartile, or below the first quartile, on any dimension.

18.6.2 Changing the Model

Outliers are points that break a pattern. This can be because the points are bad, or because we made a bad guess about the pattern. Figure 18.7 shows data where the cloud of points on the right are definite outliers for any linear model. But I drew those points following a quadratic model, and they fall perfectly along it (as they should). Deleting them, in order to make a linear model work better, would have been short-sighted at best.

The moral of Figure 18.7 is that data points can look like outliers because we're looking for the wrong pattern. If when we find apparent outliers and we can't convince ourselves that data is erroneous or irrelevant, we should consider changing our model, before, or as well as, deleting them.

18.6.3 Robust Linear Regression

A final alternative is to change how we estimate our model. Everything we've done has been based on ordinary least-squares (OLS) estimation. Because the squared error grows very rapidly with the error, OLS can be very strongly influenced by a few large "vertical" errors[10]. We might, therefore, consider using not a different statistical model, but a different method of estimating its parameters. Estimation techniques which are less influenced by outliers in the residuals than OLS are called **robust estimators**, or (for regression models) **robust regression**.

Usually (though not always), robust estimation, like OLS, tries to minimize[11] some average of a function of the errors:

$$\tilde{\beta} = \operatorname*{argmin}_{\mathbf{b}} \frac{1}{n}\sum_{i=1}^{n} \rho(y_i - \mathbf{x}_i\mathbf{b}) \tag{18.41}$$

Different choices of ρ, the **loss function**, yield different estimators. $\rho(u) = |u|$ is **least absolute deviation** (LAD) estimation[12]. $\rho(u) = u^2$ is OLS again. A popular compromise is to use **Huber's** loss function[13]

$$\rho(u) = \begin{cases} u^2 & |u| \leq c \\ 2c|u| - c^2 & |u| \geq c \end{cases} \tag{18.42}$$

Notice that Huber's loss looks like squared error for small errors, but like absolute error for large errors[14]. Huber's loss is designed to be continuous at c, and have a continuous first derivative there as well (which helps with optimization). We need to pick the scale c at which it switches over from acting like squared error to acting

[10]Suppose there are 100 data points, and we start with parameter values where $e_1 > 10$, while e_2 through $e_{100} = 0$. Changing to a new parameter value where $e_i = 1$ for all i actually reduces the MSE, even though it moves us away from perfectly fitting 99% of the data points.

[11]Hence the name "M-estimators".

[12]For minimizing absolute error, the scenario suggested in the previous footnote seems like a horrible idea, the average loss function goes from 0.1 to 1.0.

[13]Often written ψ, since that's the symbol Huber used when he introduced it. Also, some people define it as 1/2 of the way I have here; this way, though, it's identical to squared error for small u.

[14]If we set $c = 1$ in our little scenario, the average loss would go from 0.19 to 1.0, a definite worsening.

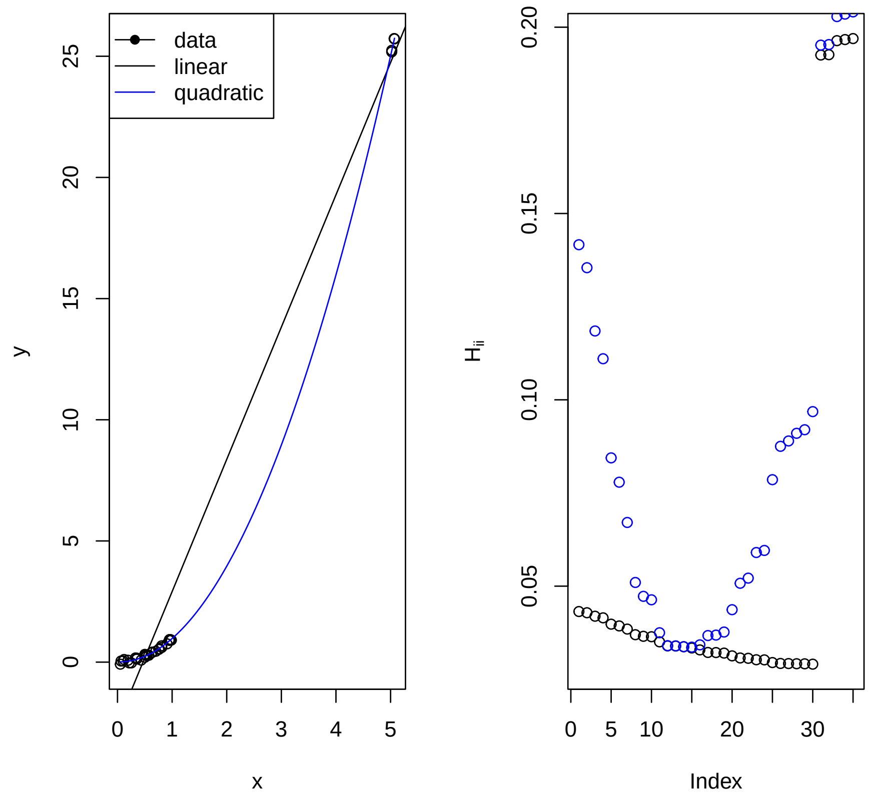


FIGURE 18.7: *The points in the upper-right are outliers for any linear model fit through the main body of points, but dominate the line because of their very high leverage; they'd be identified as outliers. But all points were generated from a quadratic model.*

like absolute error; this is usually done using a robust estimate of the noise standard deviation σ.

Robust estimation with Huber's loss can be conveniently done with the `rlm` function in the `MASS` package, which, as the name suggests, is designed to work very much like `lm`.

```
library(MASS)
summary(rlm(Mobility ~ Commute, data = mobility))
##
## Call: rlm(formula = Mobility ~ Commute, data = mobility)
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.148719 -0.019461 -0.002341  0.021093  0.332347
##
## Coefficients:
##             Value   Std. Error t value
## (Intercept)  0.0028  0.0043     0.6398
## Commute      0.2077  0.0091    22.7939
##
## Residual standard error: 0.0293 on 727 degrees of freedom
```

Robust linear regression is designed for the situation where it's still true that $Y = X\beta + \epsilon$, but the noise ϵ is not very close to Gaussian, and indeed is sometimes "contaminated" by wildly larger values. It does nothing to deal with non-linearity, or correlated noise, or even some points having excessive leverage because we're insisting on a linear model.

18.7 Exercises

1. Prove that in a simple linear regression

$$H_{ii} = \frac{1}{n}\left(1 + \frac{(x_i - \overline{x})^2}{s_X^2}\right) \tag{18.43}$$

2. Show that $(\mathbf{I} - \mathbf{H})\mathbf{x}\mathbf{c} = 0$ for any matrix $\mathbf{c}$.

3. Every row of the hat matrix has entries that sum to 1.

   (a) Show that if all of the y_i are equal, say c, then $\hat{\beta}_0 = c$ and all the estimated slopes are 0.

   (b) Using the previous part, show that $\mathbf{1}$, the $n \times 1$ matrix of all 1s, must be an eigenvector of the hat matrix with eigenvalue 1, $\mathbf{H}\mathbf{1} = \mathbf{1}$.

   (c) Using the previous part, show that the sum of each row of $\mathbf{H}$ must be 1, $\sum_{j=1}^{n} H_{ij} = 1$ for all i.

4. *Fitted values after deleting a point*

(a) (Easier) Presume that $\mathbf{H}^{(-i)}$ can be found by setting $\mathbf{H}^{(-i)}_{jk} = \mathbf{H}_{jk}/(1 - \mathbf{H}_{ji})$. Prove Eq. 18.30.

(b) (Challenging) Let $\mathbf{x}^{(-i)}$ be $\mathbf{x}$ with its i^{th} row removed. By construction, $\mathbf{H}^{(-i)}$, the $n \times (n-1)$ matrix which gives predictions at all of the original data points, is

$$\mathbf{H}^{(-i)} = \mathbf{x}((\mathbf{x}^{(-i)})^T \mathbf{x}^{(-i)})^{-1} (\mathbf{x}^{(-i)})^T \quad (18.44)$$

Show that this matrix has the form claimed in the previous problem.

5. (Challenging) Derive Eq. 18.38 for Cook's statistic from the definition. *Hint:* First, derive a formula for $\widehat{\mathbf{m}}^{(-i)}_j$ in terms of the hat matrix. Next, substitute in to the definition of D_i. Finally, you will need to use properties of the hat matrix to simplify.

Chapter 19

Model Selection

19.1 Generalization and Optimism

We estimated our model by minimizing the mean squared error on our data:

$$\widehat{\beta} = \operatorname*{argmin}_{\mathbf{b}} \frac{1}{n}(\mathbf{y}-\mathbf{x}\mathbf{b})^T(\mathbf{y}-\mathbf{x}\mathbf{b}) \tag{19.1}$$

Different linear models amount to different choices of the design matrix $\mathbf{x}$ — we add or drop variables, we add or drop interactions or polynomial terms, etc., and this adds or removes columns from the design matrix. We might consider selecting among models themselves by minimizing the MSE. This is a very bad idea, for a fundamental reason:

Every model is too optimistic about how well it will actually predict.

Let's be very clear about what it would mean to predict well. The most challenging case would be that we see a new *random* point, with predictor values $X_1, \ldots X_p$ and response Y, and our *old* $\widehat{\beta}$ has a small expected squared error:

$$\mathbb{E}\left[\left(Y - \left(\hat{\beta}_0 + \sum_{j=1}^{p} X_j \hat{\beta}_j\right)\right)^2\right] \tag{19.2}$$

Here both Y and the X's are random (hence the capital letters), so we might be asking the model for a prediction at a point it never saw before. (Of course if we have multiple identically distributed (X, Y) pairs, the expected MSE over those points is just the same as the expected squared error at one point.)

An easier task would be to ask the model for predictions at the *same* values of the predictor variables as before, but with different random noises. That is, we fit the model to

$$\mathbf{Y} = \mathbf{x}\beta + \epsilon \tag{19.3}$$

and now Tyche[1] reach into her urn and gives us

$$\mathbf{Y}' = \mathbf{x}\beta + \epsilon' \tag{19.4}$$

where ϵ and ϵ' are independent but identically distributed. The design matrix is the same, the true parameters β are the same, but the noise is different[2]. We now want to see if the coefficients we estimated from $(\mathbf{x}, \mathbf{Y})$ can predict $(\mathbf{x}, \mathbf{Y}')$. Since the only thing that's changed is the noise, if the coefficients can't predict well any more, that means that they were really just memorizing the noise, and not actually doing anything useful.

Our out-of-sample expected MSE, then, is

$$\mathbb{E}\left[n^{-1}(\mathbf{Y}' - \mathbf{x}\widehat{\beta})^T(\mathbf{Y}' - \mathbf{x}\widehat{\beta})\right] \tag{19.5}$$

It will be convenient to break this down into an average over data points, and to abbreviate $\mathbf{x}\widehat{\beta} = \widehat{\mathbf{m}}$, the vector of fitted values. Notice that since the predictor variables and the coefficients aren't changing, our predictions are the same both in and out of sample — at point i, we will predict $\widehat{m}_i$.

In this notation, then, the expected out-of-sample MSE is

$$\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}(Y_i' - \widehat{m}_i)^2\right] \tag{19.6}$$

We'll compare this to the expected in-sample MSE,

$$\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}(Y_i - \widehat{m}_i)^2\right] \tag{19.7}$$

Notice that $\widehat{m}_i$ is a function of Y_i (among other things), so those are dependent random variables, while $\widehat{m}_i$ and Y_i' are completely statistically independent[3].

Break this down term by term. What's the expected value of the i^{th} in-sample squared error?

$$\begin{aligned}
&\mathbb{E}\left[(Y_i - \hat{m}_i)^2\right] && (19.8)\\
&= \operatorname{Var}[Y_i - \hat{m}_i] + (\mathbb{E}[Y_i - \hat{m}_i])^2 &&\\
&= \operatorname{Var}[Y_i] + \operatorname{Var}[\hat{m}_i] - 2\operatorname{Cov}[Y_i, \hat{m}_i] + (\mathbb{E}[Y_i] - \mathbb{E}[\hat{m}_i])^2 && (19.9)
\end{aligned}$$

The covariance term is not (usually) zero, because, as I just said, $\hat{m}_i$ is a function of, in part, Y_i.

[1]Look her up.

[2]If we really are in an experimental setting, we really could get a realization of $\mathbf{Y}'$ just by running the experiment a second time. With surveys or with observational data, it would be harder to actually realize $\mathbf{Y}'$, but mathematically at least it's unproblematic.

[3]That might sound weird, but remember we're holding $\mathbf{x}$ fixed in this exercise, so what we mean is that knowing $\widehat{m}_i$ doesn't give us an extra information about Y_i' beyond what we'd get from knowing the values of the X variables.

On the other hand, what's the expected value of the i^{th} squared error on new data?

$$\mathbb{E}\left[(Y_i' - \hat{m}_i)^2\right] \tag{19.10}$$
$$= \mathrm{Var}\left[Y_i\prime - \hat{m}_i\right] + \left(\mathbb{E}\left[Y_i' - \hat{m}_i\right]\right)^2$$
$$= \mathrm{Var}\left[Y_i'\right] + \mathrm{Var}\left[\hat{m}_i\right] - 2\mathrm{Cov}\left[Y_i', \hat{m}_i\right] + \left(\mathbb{E}\left[Y_i'\right] - \mathbb{E}\left[\hat{m}_i\right]\right)^2 \tag{19.11}$$

Y_i' is independent of Y_i, but has the same distribution. This tells us that $\mathbb{E}\left[Y_i'\right] = \mathbb{E}\left[Y_i\right]$, $\mathrm{Var}\left[Y_i'\right] = \mathrm{Var}\left[Y_i\right]$, but $\mathrm{Cov}\left[Y_i', \hat{m}_i\right] = 0$. So

$$\mathbb{E}\left[(Y_i' - \hat{m}_i)^2\right] = \mathrm{Var}\left[Y_i\right] + \mathrm{Var}\left[\hat{m}_i\right] + (\mathbb{E}\left[Y_i\right] - \mathbb{E}\left[\hat{m}_i\right])^2 \tag{19.12}$$
$$= \mathbb{E}\left[(Y_i - \hat{m}_i)^2\right] + 2\mathrm{Cov}\left[Y_i, \hat{m}_i\right] \tag{19.13}$$

Averaging over data points,

$$\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}(Y_i' - \widehat{m}_i)^2\right] = \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}(Y_i - \widehat{m}_i)^2\right] + \frac{2}{n}\sum_{i=1}^{n}\mathrm{Cov}\left[Y_i, \hat{m}_i\right] \tag{19.14}$$

Clearly, we need to get a handle on that sum of covariances.

For a linear model, though, $\mathrm{Cov}\left[Y_i, \hat{m}_i\right] = \sigma^2 H_{ii}$ (Exercise **??**). So, for linear models,

$$\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}(Y_i' - \widehat{m}_i)^2\right] = \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}(Y_i - \widehat{m}_i)^2\right] + \frac{2}{n}\sigma^2 \operatorname{tr}\mathbf{H} \tag{19.15}$$

and we know that with p predictors and one intercept, $\operatorname{tr}\mathbf{H} = p + 1$ (Exercise 6 in Chapter 12). Thus, for linear models,

$$\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}(Y_i' - \widehat{m}_i)^2\right] = \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}(Y_i - \widehat{m}_i)^2\right] + \frac{2}{n}\sigma^2(p+1) \tag{19.16}$$

Of course, we don't actually know the *expectation* on the right-hand side, but we do have a sample estimate of it, which is the in-sample MSE. If the law of large numbers is still our friend,

$$\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}(Y_i' - \widehat{m}_i)^2\right] \approx \frac{1}{n}\sum_{i=1}^{n}(Y_i - \widehat{m}_i)^2 + \frac{2}{n}\sigma^2(p+1) \tag{19.17}$$

The second term on the right, $(2/n)\sigma^2(p+1)$, is the **optimism** of the model — the amount by which its in-sample MSE systematically under-estimates its true expected squared error. Notice that this:

- Grows with σ^2: more noise gives the model more opportunities to seem to fit well by capitalizing on chance.
- Shrinks with n: at any fixed level of noise, more data makes it harder to pretend the fit is better than it really is.

- Grows with p: every extra parameter is another control which can be adjusted to fit to the noise.

Minimizing the in-sample MSE completely ignores the bias from optimism, so it is guaranteed to pick models which are too large and predict poorly out of sample. If we could calculate the optimism term, we could at least use an unbiased estimate of the true MSE on new data.

Of course, we do not actually know σ^2.

19.2 Mallow's C_p Statistic

The Mallows C_p statistic just substitutes in a feasible estimator of σ^2, which is $\hat{\sigma}^2$ *from the largest model we consider*. This will be an unbiased estimator of σ^2 if the real model is smaller (contains a strict subset of the predictor variables), but not vice versa[4].

That is, for a linear model with $p+1$ coefficients fit by OLS,

$$C_p \equiv \frac{1}{n}\sum_{i=1}^{n}(Y_i - \widehat{m}_i)^2 + \frac{2}{n}\hat{\sigma}^2(p+1) \tag{19.18}$$

The selection rule is to pick the model which minimizes C_p.

We can think of C_p as having two parts,

$$C_p = MSE + (\text{penalty}) \tag{19.19}$$

From one point of view, the penalty is just an estimate of the bias. From another point of view, it's a cost we're imposing on models for having extra parameters. Every new parameter has got to pay that cost by reducing the MSE by at least a certain amount; if it doesn't, the extra parameter isn't worth it.

(Before this, we've only been dealing with *one* model, so we've not had to distinguish carefully between the in-sample MSE and the maximum likelihood estimate of σ^2. With multiple models floating around, though, each can have its own MSE, but there is only one true σ^2, and we need *an* estimate of it.)

For comparing models, we really care about differences:

$$\Delta C_p = MSE_1 - MSE_2 + \frac{2}{n}\hat{\sigma}^2(p_1 - p_2) \tag{19.20}$$

(The extra term for the intercept, being common to both models, doesn't contribute.)

Alternate form of C_p You will find many references which define C_p somewhat differently:

$$\frac{nMSE}{\hat{\sigma}^2} - n + 2p \tag{19.21}$$

[4]This assumes the largest model must contain the truth!

and say that the optimal value is close to p, not close to 0. To see that this selects exactly the same models as the rule given above, take a difference between two models, with MSE's MSE_1, MSE_2 and p_1, p_2 predictors. We get

$$\frac{n(MSE_1 - MSE_2)}{\hat{\sigma}^2} + 2(p_1 - p_2) \tag{19.22}$$

Dividing by n and multiplying by $\hat{\sigma}^2$ gives us back Eq. 19.20. There are reasons to assert that Eq. 19.21 should indeed be close to p for the right model (if the Gaussian noise assumption holds), but Eq. 19.18 is a good estimate of the out-of-sample error, and a good model selection rule, much more broadly.

19.2.1 R^2 and Adjusted R^2

Recall that

$$R^2 = 1 - \frac{MSE}{s_Y^2} \tag{19.23}$$

Picking a model by maximizing R^2 is thus equivalent to picking a model by minimizing MSE. It is therefore stupid for exactly the same reasons that minimizing MSE across models is stupid.

Recall that the adjusted R^2 is

$$R^2_{adj} = 1 - \frac{MSE\frac{n}{n-p-1}}{s_Y^2} \tag{19.24}$$

That is, it's R^2 with the unbiased estimator of σ^2. Maximizing adjusted R^2 therefore corresponds to minimizing that unbiased estimator. What does that translate to?

$$\begin{aligned} MSE\frac{n}{n-p-1} &= MSE\frac{1}{1-(p+1)/n} && (19.25)\\ &\approx MSE\left(1+\frac{p+1}{n}\right) && (19.26)\\ &= MSE + MSE\frac{p+1}{n} && (19.27)\end{aligned}$$

where the approximation becomes exact as $n \to \infty$ with p fixed[5]. Even for the completely right model, where MSE is a consistent estimator of $\hat{\sigma}^2$, the correction or penalty is only half as big as we've seen it should be. Selecting models using adjusted R^2 is not completely stupid, as maximizing R^2 is, but it is still not going to work very well.

[5]Use the binomial theorem to expand $1/(1-u)$ as $1+u+u^2+\ldots$, and truncate the series at first order. (If u is small, u^2 is tiny, and the higher powers microscopic.)

19.3 Akaike Information Criterion (AIC)

The great Japanese statistician Hirotugu Akaike proposed a famous model selection rule which also has the form of "in-sample performance plus penalty". What has come to be called the **Akaike information criterion** (AIC) is

$$AIC(S) \equiv L_S - \dim(S) \tag{19.28}$$

where L_S is the log likelihood of the model S, evaluated at the maximum likelihood estimate, and $\dim(S)$ is the dimension of S, the number of adjustable parameters it has. Akaike's rule is to pick the model which maximizes AIC[6].

The reason for this definition is that Akaike showed AIC/n is an unbiased estimate of the expected log-probability the estimated parameters will give to a new data point which it hasn't seen before, if the model is right. This is the natural counterpart of expected squared error for more general distributions than the Gaussian. If we do specialize to linear-Gaussian models, then we've seen (Chapter 10) that

$$L = -\frac{n}{2}(1+\log 2\pi) - \frac{n}{2}\log MSE \tag{19.29}$$

and the dimension of the model is $p+2$ (because σ^2 is also an adjustable parameter). Notice that $-\frac{n}{2}(1+\log 2\pi)$ doesn't involve the parameters at all. If we compare AICs for two models, with mean squared errors in-sample of MSE_1 and MSE_2, and one with p_1 predictors and the other with p_2, the difference in AICs will be

$$\Delta AIC = -\frac{n}{2}\log MSE_1 + \frac{n}{2}\log MSE_2 - (p_1 - p_2) \tag{19.30}$$

To relate this to C_p, let's write $MSE_2 = MSE_1 + \Delta MSE$. Then

$$\begin{aligned} \Delta AIC &= -\frac{n}{2}\log MSE_1 + \frac{n}{2}\log MSE_1\left(1+\frac{\Delta MSE}{MSE_1}\right) - (p_1-p_2) && (19.31)\\ &= -\frac{n}{2}\log\left(1+\frac{\Delta MSE}{MSE_1}\right) - (p_1-p_2) && (19.32)\end{aligned}$$

Now let's suppose that model 1 is actually the correct model, so $MSE_1 = \hat{\sigma}^2$, and that ΔMSE is small compared to $\hat{\sigma}^2$, so[7]

$$\begin{aligned} \Delta AIC &\approx -\frac{n}{2}\frac{\Delta MSE}{\hat{\sigma}^2} - (p_1-p_2) && (19.33)\\ \frac{-2\hat{\sigma}^2}{n}\Delta AIC &\approx \Delta MSE + \frac{2}{n}\hat{\sigma}^2(p_1-p_2) = \Delta C_p && (19.34)\end{aligned}$$

So, if one of the models we're looking at is actually the correct model, and the others aren't too different from it, picking by maximizing AIC will give the same answer as picking by minimizing C_p.

[6]Actually, in his original paper (Akaike, 1973), he proposed using *twice* this, to simplify some calculations involving chi-squared distributions. Many subsequent authors have since kept the factor of 2, which of course will not change which model is selected. Also, some authors define AIC as negative of this, and then minimize it; again, clearly the same thing.

[7]Taylor expand $\log 1+u$ around 1 to get $\log 1+u \approx u$, for u close to 0.

Other Uses of AIC AIC can be applied whenever we have a likelihood. It is therefore used for tasks like comparing models of probability distributions, or predictive models where the whole distribution is important. C_p, by contrast, really only makes sense if we're trying to do regression and want to use squared error.

19.3.1 Why $-\dim(S)$?

Akaike had a truly brilliant argument for subtracting a penalty equal to the number of parameters from the log-likelihood, which is too pretty not to at least sketch here.[8]

Generically, say that the parameter vector is θ, and its true value is θ^*. (For linear regression with Gaussian noise, θ consists of all $p+1$ coefficients plus σ^2.) The length of this vector, which is $\dim(S)$, is let's say d. (For linear regression with Gaussian noise, $d = p+2$.) The maximum likelihood estimate is $\hat{\theta}$. We know that the derivative of the likelihood is zero at the MLE:

$$\nabla L(\hat{\theta}) = 0 \tag{19.35}$$

Let's do a Taylor series expansion of $\nabla L(\theta)$ around the true parameter value θ^*:

$$\nabla L(\theta) = \nabla L(\theta^*) + (\theta - \theta^*)\nabla\nabla L(\theta^*) \tag{19.36}$$

Here $\nabla\nabla L(\theta^*)$ is the $d \times d$ matrix of second partial derivatives of L, evaluated at θ^*. This is called the **Hessian**, and would traditionally be written $\mathbf{H}$, but that would lead to confusion with the hat matrix, so I'll call it $\mathbf{K}$. Therefore the Taylor expansion for the gradient of the log-likelihood is

$$\nabla L(\theta) = \nabla L(\theta^*) + (\theta - \theta^*)\mathbf{K} \tag{19.37}$$

Applied to the MLE,

$$0 = \nabla L(\theta^*) + (\hat{\theta} - \theta^*)\mathbf{K} \tag{19.38}$$

or

$$\hat{\theta} = \theta^* - K^{-1}\nabla L(\theta^*) \tag{19.39}$$

What is the *expected* log-likelihood, on new data, of $\hat{\theta}$? Call this expected log-likelihood ℓ (using a lower-case letter to indicate that it is non-random). Doing another Taylor series,

$$\ell(\theta) \approx \ell(\theta^*) + (\theta - \theta^*)^T\nabla\ell(\theta^*) + \frac{1}{2}(\theta - \theta^*)^T\nabla\nabla\ell(\theta^*)(\theta - \theta^*) \tag{19.40}$$

However, it's not hard to show that the expected log-likelihood is always[9] maximized by the true parameters, so $\nabla\ell(\theta^*) = 0$. (The same argument also shows $\mathbb{E}[\nabla L(\theta^*)] = 0$.) Call the Hessian in this Taylor expansion $\mathbf{k}$. (Again, notice the lower-case letter for a non-random quantity.) We have

$$\ell(\theta) \approx \ell(\theta^*) + \frac{1}{2}(\theta - \theta^*)^T\mathbf{k}(\theta - \theta^*) \tag{19.41}$$

[8] Nonetheless, this subsection is optional.

[9] Except for quite weird models.

Apply this to the MLE:

$$\ell(\hat{\theta}) \approx \ell(\theta^*) + \frac{1}{2}\nabla L(\theta^*)\mathbf{K}^{-1}\mathbf{k}\mathbf{K}^{-1}\nabla L(\theta^*) \tag{19.42}$$

Taking expectations,

$$\mathbb{E}\left[\ell(\hat{\theta})\right] \approx \ell(\theta^*) + \frac{1}{2}\operatorname{tr}\mathbf{K}^{-1}\mathbf{k}\mathbf{K}^{-1}\mathbf{J} \tag{19.43}$$

where $\operatorname{Var}[\nabla L(\theta^*)] = \mathbf{J}$. For large n, $\mathbf{K}$ converges on $\mathbf{k}$, so this simplifies to

$$\mathbb{E}\left[\ell(\hat{\theta})\right] \approx \ell(\theta^*) + \frac{1}{2}\operatorname{tr}\mathbf{k}^{-1}\mathbf{J} \tag{19.44}$$

This still leaves things in terms of $\ell(\theta^*)$, which of course we don't know, but now we do another Taylor expansion, this time of L around $\hat{\theta}$:

$$L(\theta^*) \approx L(\hat{\theta}) + \frac{1}{2}(\theta^* - \hat{\theta})^T \nabla\nabla L(\hat{\theta})(\theta^* - \hat{\theta}) \tag{19.45}$$

so

$$L(\theta^*) \approx L(\hat{\theta}) + \frac{1}{2}(\mathbf{K}^{-1}\nabla L(\theta^*))^T \nabla\nabla L(\hat{\theta})(\mathbf{K}^{-1}\nabla L(\theta^*)) \tag{19.46}$$

For large n, $\nabla\nabla L(\hat{\theta}) \to \nabla\nabla L(\theta^*) \to \mathbf{k}$. So, again taking expectations,

$$\ell(\theta^*) \approx \mathbb{E}\left[L(\hat{\theta}\right] + \frac{1}{2}\operatorname{tr}\mathbf{k}^{-1}\mathbf{J} \tag{19.47}$$

Putting these together,

$$\mathbb{E}\left[\ell(\hat{\theta})\right] \approx \mathbb{E}\left[L(\hat{\theta}\right] + \operatorname{tr}\mathbf{k}^{-1}\mathbf{J} \tag{19.48}$$

An unbiased estimate is therefore

$$L(\hat{\theta}) + \operatorname{tr}\mathbf{k}^{-1}\mathbf{J} \tag{19.49}$$

Finally, a fundamental result (the "Fisher identity") says that for well-behaved models, *if* the model is correct, then

$$\operatorname{Var}[\nabla L(\theta^*)] = -\nabla\nabla\ell(\theta^*) \tag{19.50}$$

or $\mathbf{J} = -\mathbf{k}$. Hence, if the model is correct, our unbiased estimate is just

$$L(\hat{\theta}) - \operatorname{tr}\mathbf{I} \tag{19.51}$$

and of course $\operatorname{tr}\mathbf{I} = d$.

There, as you'll notice, several steps where we're making a bunch of approximations. Some of these approximations (especially those involving the Taylor expansions) can be shown to be OK asymptotically (i.e., as $n \to \infty$) by more careful math. The last steps, however, where we invoke the Fisher identity, are rather more dubious. (After all, all of the models we're working with can hardly contain the true distribution.) A somewhat more robust version of AIC is therefore to use as the criterion

$$L(\hat{\theta}) + \operatorname{tr}\mathbf{KJ} \tag{19.52}$$

19.4 Leave-one-out Cross-Validation (LOOCV)

When looking at influential points and outliers, we considered omitting one point from the data set, estimating the model, and then trying to predict that one data point. The **leave-one-out** fitted value for data point i is $\hat{m}_i^{(-i)}$, where the subscript $(-i)$ indicates that point i was left out in calculating this fit. The **leave-one-out cross-validation score** of the model is

$$LOOCV = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{m}_i^{(-i)})^2 \tag{19.53}$$

(Many more old-fashioned regression textbooks look at $nLOOCV$, and call it PRESS, "predictive residual sum of squares".)

The story for cross-validation is pretty compelling: we want to know if our model can generalize to new data, so *see* how well it generalizes to new data. Leaving out each point in turn ensures that that the set of points on which we try to make predictions is just as representative of the whole population as the original sample was. Fortunately, this is one of those cases where a compelling story is actually true: LOOCV is an unbiased estimate of the generalization error.

19.4.1 Short-cut Based on Leverage

Re-estimating the model n times would be seriously time-consuming, but there is fortunately a short-cut:

$$LOOCV = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{Y_i - \hat{m}_i}{1-H_{ii}}\right)^2 \tag{19.54}$$

The numerator inside the square is just the residual of the model fit to the full data. This gets divided by $1-H_{ii}$, which is also something we can calculate with just one fit to the model. (The denominator says that the residuals for high-leverage points count more, and those for low-leverage points count less. If the model is going out of its way to match Y_i (high leverage H_{ii}) and it still can't fit it, that's worse than the same sized residual at a point the model doesn't really care about (low leverage).)

The gap between LOOCV and the MSE can be thought of as a penalty, just like with C_p or AIC. The penalty doesn't have such a nice mathematical expression, but it's well-defined and easy for us to calculate.

It also converges to the penalty C_p applies as n grows. To help see this, first observe that the H_{ii} must be getting small. (We know that $\sum_i H_{ii} = p+1$.) Then[10] $(1-H_{ii})^{-2} \approx 1-2H_{ii}$, and

$$LOOCV \approx \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{m}_i)^2(1-2H_{ii}) \approx MSE + 2\sigma^2 \operatorname{tr}\mathbf{H} \tag{19.55}$$

[10]Use the binomial theorem again.

Cross-validation with log-likelihood The leave-one-out idea can also be applied for any model where we make a probabilistic prediction. Instead of measuring mean squared error, we measure the negative log probability density the model assigns to the actual left-out point. (Negative, so that a lower score is still better.) With Gaussian noise, this comes to the same thing as the MSE, of course.

19.4.2 Summing Up C_p, AIC, LOOCV

Under a very broad range of circumstances, there are theorems which say, roughly, the following:

> As $n \to \infty$, the expected out-of-sample MSE of the model picked by leave-one-out cross-validation is close to that of the best model considered.

The condition for these results do *not* require that any of the models considered be true, or that the true model have Gaussian noise or even be linear.

As we've seen, for large n leave-one-out and Mallow's C_p become extremely similar, and will pick the same model, and so will AIC, if one of the models is right. So they will also pick models which predict almost as well as the best of the models we're working with. Since C_p and AIC involve less calculation than leave-one-out, they have advantages when n is large. Against this, there don't seem to be any situations where C_p or AIC pick models with good predictive performance but leave-one-out does not. The best way to think about C_p and AIC is that they are fast approximations to the more fundamental quantity, which is leave-one-out.

On the other hand, one can *also* prove the following:

> As $n \to \infty$, if the true model is among those being compared, LOOCV, C_p and AIC will all tend to pick a *strictly larger* model than the truth.

That is, all three criteria tend to prefer models which are bigger than the true model, even when the true model is available to them. They are "not consistent for model selection".

The problem is that while these methods give unbiased estimates of the generalization error, that doesn't say anything about the variance of the estimates. Models with more parameters have higher variance, and the penalty applied by these methods isn't strong enough to overcome the chance of capitalizing on that variance.

19.5 Other Model Selection Criteria

While many, many other model selection criteria have been proposed, two are particularly important.

19.5.1 *k*-Fold Cross-Validation

In leave-one-out cross-validation, we omitted each data point in turn, and tried to predict it. *K*-fold cross-validation is somewhat different, and goes as follows.

- Randomly divide the data into k equally-sized parts, or "folds".
- For each fold
  - Temporarily hold back that fold, calling it the "testing set".
  - Call the other $k-1$ folds, taken together, the "training set".
  - Estimate each model on the training set.
  - Calculate the MSE of each model on the testing set.
- Average MSEs over folds.

We then pick the model with the lowest MSE, averaged across testing sets.

The point of this is just like the point of leave-one-out: the models are compared only on data which they didn't get to see during estimation. Indeed, leave-one-out is the special case of k-fold cross-validation where $k = n$. The disadvantage of doing that is that in leave-one-out, all of the training sets are very similar (they share $n-2$ data points), so averaging over folds does very little to reduce variance. For moderate k — people typically use 5 or 10 — k-fold CV tends to produce very good model selection results.

Like leave-one-out CV, k-fold cross-validation can be applied to any loss function, such as the proportion of cases mis-classified, or negative log-likelihood.

19.5.2 BIC

A more AIC-like criterion is the "Bayesian[11] information criterion" introduced by Schwarz (1978). The name is quite misleading[12], but irrelevant; it's got the exact same idea of penalizing the log-likelihood with the number of parameters, but using a penalty which gets bigger with n:

$$BIC(S) = L_S - \frac{\log n}{2}\dim(S) \tag{19.56}$$

This is a stronger penalty than AIC applies, and this has consequences:

> As $n \rightarrow \infty$, if the true model is among those BIC can select among, BIC will tend to pick the true model.

Of course there are various conditions attached to this, some of them quite technical, but it's generally true for IID samples, for regression modeling, for many sorts of time series model, etc. Unfortunately, the model selected by BIC will tend to predict less well than the one selected by leave-one-out cross-validation or AIC.

[11]Bayesianism is the idea that we ought to have probabilities for parameter values and for models, and not just for random variables (or, said another way, to treat parameters and models as also random variables), and update those probabilities as we see more events using Bayes's rule. It is a controversial position within statistics and philosophy of science, with many able and learned supporters, and equally able and learned opponents. (It is also the only position in statistics and philosophy of science I know of which has an online cult dedicated to promoting it, alongside reading certain works of Harry Potter fanfic, and trying not to think about the possibility a future superintelligent computer will simulate your being tortured.)

[12]The truly Bayesian position is not to *select* a model at all, but rather to maintain a probability distribution over all models you think possible.

19.6 Stepwise Model Selection

One way to automatically select a model is to begin with the largest model you can, and then prune it, which can be done in several ways:

- Eliminate the least-significant coefficient.
- Pick your favorite model selection criterion, consider deleting each coefficient in turn, and pick the sub-model with the best value of the criterion.

Having eliminated a variable, one then re-estimates the model, and repeats the procedure. Stop when either all the remaining coefficients are significant (under the first option), or nothing can be eliminated without worsening the criterion.

(What I've described is **backwards** stepwise model selection. **Forward** stepwise model selection starts with the intercept-only model and adds variables in the same fashion. There are, naturally, forward-backward hybrids.)

Stepwise model selection is a **greedy** procedure: it takes the move which does the most to immediately improve the criterion, without considering the consequences down the line. There are very, very few situations where it is consistent for model selection, or (in its significance-testing version) where it even does a particularly good job of coming up with predictive models, but it's surprisingly popular.

19.7 Inference after Selection

All of the inferential statistics we have done in earlier chapters presumed that our choice of model was completely fixed, and not at all dependent on the data. If different data sets would lead us to use different models, and our data are (partly) random, then which model we're using is also random. This leads to some extra uncertainty in, say, our estimate of the slope on X_1, which is *not* accounted for by our formulas for the sampling distributions, hypothesis tests, confidence sets, etc.

A very common response to this problem, among practitioners, is to ignore it, or at least hope it doesn't matter. This can be OK, if the data-generating distribution forces us to pick one model with very high probability, or if all of the models we might pick are very similar to each other. Otherwise, ignoring it leads to nonsense.

Here, for instance, I simulate 200 data points where the Y variable is a standard Gaussian, and there are 100 independent predictor variables, all also standard Gaussians, independent of each other *and of* Y:

Chapter 20

Review

Let us review[1].

20.1 Basic Model Assumptions (without Gaussian Noise)

We model one continuous response variable, as a linear function of p numerical predictors, plus noise:

$$Y = \beta_0 + \beta_1 X_1 + \ldots \beta_p X_p + \epsilon \tag{20.1}$$

Linearity is an assumption, which can be wrong.

Further assumptions take the form of restrictions on the noise:

$$\begin{aligned} \mathbb{E}[\epsilon|X] &= 0 && (20.2)\\ \mathrm{Var}[\epsilon|X] &= \sigma^2 && (20.3)\\ & && (20.4) \end{aligned}$$

Moreover, we assume ϵ is uncorrelated across observations.

We convert this to matrix form:

$$\mathbf{Y} = \mathbf{x}\beta + \epsilon \tag{20.5}$$

$\mathbf{Y}$ is an $n \times 1$ matrix of random variables; $\mathbf{x}$ is an $n \times (p+1)$ matrix, with an extra column of all 1s; ϵ is an $n \times 1$ matrix. Beyond linearity, the assumptions translate to

$$\mathbb{E}[\epsilon|\mathbf{x}] = 0 \tag{20.6}$$

and

$$\mathrm{Var}[\epsilon|\mathbf{x}] = \sigma^2\mathbf{I} \tag{20.7}$$

We don't know β. If we guess it is $\mathbf{b}$, we will make an $n \times 1$ vector of predictions

$$\mathbf{xb} \tag{20.8}$$

[1]All of the theory for simple linear regression is a special case of what follows, with $p = 1$. The entire first third of the course was just a warm-up period.

and have an $n \times 1$ vector of errors

$$\mathbf{y} - \mathbf{x}\mathbf{b} \tag{20.9}$$

The mean squared error, as a function of $\mathbf{b}$, is then

$$MSE(\mathbf{b}) = \frac{1}{n}(\mathbf{y} - \mathbf{x}\mathbf{b})^T(\mathbf{y} - \mathbf{x}\mathbf{b}) \tag{20.10}$$

20.2 Least Squares Estimation and Its Properties

The least squares estimate of the coefficients is the one which minimizes the MSE:

$$\widehat{\beta} \equiv \underset{\mathbf{b}}{\operatorname{argmin}} MSE(\mathbf{b}) \tag{20.11}$$

To find this, we need the derivatives:

$$\nabla_{\mathbf{b}} MSE = \frac{2}{n}(\mathbf{x}^T\mathbf{y} - \mathbf{x}^T\mathbf{x}\mathbf{b}) \tag{20.12}$$

We set the derivative to zero at the optimum:

$$\frac{1}{n}\mathbf{x}^T\left(\mathbf{y} - \mathbf{x}\widehat{\beta}\right) = 0 \tag{20.13}$$

The term in parentheses is the vector of errors when we use the least-squares estimate. This is the vector of residuals,

$$\mathbf{e} \equiv \mathbf{y} - \mathbf{x}\widehat{\beta} \tag{20.14}$$

so the have the **normal**, **estimating** or **score** equations,

$$\frac{1}{n}\mathbf{x}^T\mathbf{e} = 0 \tag{20.15}$$

We say "equations", plural, because this is equivalent to the set of $p + 1$ equations

$$\frac{1}{n}\sum_{i=1}^{n} e_i = 0 \tag{20.16}$$

$$\frac{1}{n}\sum_{i=1}^{n} e_i x_{ij} = 0 \tag{20.17}$$

(Many people omit the factor of $1/n$.) This tells us that while $\mathbf{e}$ is an n-dimensional vector, it is subject to $p + 1$ linear constraints, so it is confined to a linear subspace of dimension $n - p - 1$. Thus $n - p - 1$ is the number of **residual degrees of freedom**.

The solution to the estimating equations is

$$\widehat{\beta} = (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{y} \tag{20.18}$$

This is one of the two most important equations in the whole subject. It says that the coefficients are a linear function of the response vector $\mathbf{y}$. Set all the responses to 0 and all the coefficients are zero; double all the responses and all the coefficients are doubled.

Say that $\mathbf{x}'$ is the $n \times p$ matrix of predictors, without the column of 1s, and β' is the $p \times 1$ vector of slopes. Finally, let $\overline{\mathbf{x}}$ be the $1 \times p$ vector of the mean values of the predictors. Then

$$\hat{\beta}_0 = \overline{y} - \overline{\mathbf{x}}\widehat{\beta'} \tag{20.19}$$

while

$$\widehat{\beta'} = \widehat{\text{Var}}[X]^{-1}\widehat{\text{Cov}}[X,Y] \tag{20.20}$$

just as in simple linear regression.

The least squares estimate is always a constant plus noise:

$$\begin{aligned}\widehat{\beta} &= (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{Y} && (20.21)\\ &= (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T(\mathbf{x}\beta+\epsilon) && (20.22)\\ &= (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{x}\beta+(\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\epsilon && (20.23)\\ &= \beta+(\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\epsilon && (20.24)\end{aligned}$$

The least squares estimate is always unbiased:

$$\mathbb{E}\left[\widehat{\beta}\right] = \beta + (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbb{E}[\epsilon] = \beta \tag{20.25}$$

Its variance always depends just on σ^2 and $\mathbf{x}^T\mathbf{x}$:

$$\text{Var}\left[\widehat{\beta}\right] = \sigma^2(\mathbf{x}^T\mathbf{x})^{-1} \tag{20.26}$$

Since the entries in $\mathbf{x}^T\mathbf{x}$ are usual proportional to n, it can be helpful to say

$$\text{Var}\left[\widehat{\beta}\right] = \frac{\sigma^2}{n}\left(\frac{1}{n}\mathbf{x}^T\mathbf{x}\right)^{-1} \tag{20.27}$$

The variance of any one coefficient estimator is

$$\text{Var}\left[\hat{\beta}_i\right] = \frac{\sigma^2}{n}\left(\frac{1}{n}\mathbf{x}^T\mathbf{x}\right)^{-1}_{i+1,i+1} \tag{20.28}$$

The vector of fitted means or conditional values is

$$\widehat{\mathbf{m}} \equiv \mathbf{x}\widehat{\beta} \tag{20.29}$$

This is more conveniently expressed in terms of the original matrices:

$$\widehat{\mathbf{m}} = \mathbf{x}(\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{y} = \mathbf{H}\mathbf{y} \tag{20.30}$$

The fitted values are thus linear in $\mathbf{y}$: set the responses all to zero and all the fitted values will be zero; double all the responses and all the fitted values will double.

The $n \times n$ **hat matrix** $H \equiv \mathbf{x}(\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T$, also called the influence, projection or prediction matrix, controls the fitted values. It is a function of $\mathbf{x}$ alone, ignoring the response variable totally. It is an $n \times n$ matrix with several important properties:

- It is symmetric, $\mathbf{H}^T = \mathbf{H}$.
- It is idempotent, $\mathbf{H}^2 = \mathbf{H}$.
- Its trace $\operatorname{tr}\mathbf{H} = \sum_i H_{ii} = p+1$, the number of degrees of freedom for the fitted values.

The variance-covariance matrix of the fitted values is

$$\operatorname{Var}[\widehat{\mathbf{m}}] = \mathbf{H}\sigma^2\mathbf{I}\mathbf{H}^T = \sigma^2\mathbf{H} \tag{20.31}$$

To make a prediction at a new point, not in the data used for estimation, we take its predictor coordinates and group them into a $1\times(p+1)$ matrix $\mathbf{x}_{new}$ (including the 1 for the intercept). The point prediction for Y is then $\mathbf{x}_{new}\widehat{\beta}$. The expected value is $\mathbf{x}_{new}\beta$, and the variance is $\operatorname{Var}\left[\mathbf{x}_{new}\widehat{\beta}\right] = \mathbf{x}_{new}\operatorname{Var}\left[\widehat{\beta}\right]\mathbf{x}_{new}^T = \sigma^2\mathbf{x}_{new}(\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}_{new}^T$.

The residuals are also linear in the response:

$$\mathbf{e} \equiv \mathbf{y} - \widehat{\mathbf{m}} = (\mathbf{I}-\mathbf{H})\mathbf{y} \tag{20.32}$$

The trace of $\mathbf{I}-\mathbf{H}$ is $n-p-1$. The variance-covariance matrix of the residuals is thus

$$\operatorname{Var}[\mathbf{e}] = \sigma^2(\mathbf{I}-\mathbf{H}) \tag{20.33}$$

The mean squared error is

$$MSE = \frac{1}{n}\sum_{i=1}^{n} e_i^2 = \frac{1}{n}\mathbf{e}^T\mathbf{e} \tag{20.34}$$

Its expectation value is slightly below σ^2:

$$\mathbb{E}[MSE] = \sigma^2\frac{n-p-1}{n} \tag{20.35}$$

(This may be proved using the trace of $\mathbf{I}-\mathbf{H}$.) An unbiased estimate of σ^2, which I will call $\hat{\sigma}^2$ throughout the rest of this, is

$$\hat{\sigma}^2 \equiv MSE\frac{n}{n-p-1} \tag{20.36}$$

The **leverage** of data point i is H_{ii}. This has several interpretations:

1. $\operatorname{Var}[\widehat{m}_i] = \sigma^2 H_{ii}$; the leverage controls how much variance there is in the fitted value.
2. $\partial\widehat{m}_i/\partial y_i = H_{ii}$; the leverage says how much changing the response value for point i changes the fitted value there.
3. $\operatorname{Cov}[\widehat{m}_i, Y_i] = \sigma^2 H_{ii}$; the leverage says how much covariance there is between the i^{th} response and the i^{th} fitted value.

4. $\text{Var}[e_i] = \sigma^2(1 - H_{ii})$; the leverage controls how big the ith residual is.

The **standardized residual** is

$$r_i = \frac{e_i}{\hat{\sigma}\sqrt{1 - H_{ii}}} \tag{20.37}$$

so all the standardized residuals have the same variance.

The only restriction we have to impose on the predictor variables X_i is that $(\mathbf{x}^T\mathbf{x})^{-1}$ needs to exist. This is equivalent to

- **x** is not **collinear**: none of its columns is a linear combination of other columns; which is also equivalent to
- The eigenvalues of $\mathbf{x}^T\mathbf{x}$ are all > 0. (If there are zero eigenvalues, the corresponding eigenvectors indicate linearly-dependent combinations of predictor variables.)

Nearly-collinear predictor variables tend to lead to large variances for coefficient estimates, with high levels of correlation among the estimates. Sometimes this uncertainty can be reduced by finding new variables which contain most (or all) of the same information, but are less correlated with each other.

It is perfectly OK for one column of **x** to be a function of another, provided it is a nonlinear function. Thus in **polynomial** regression we add extra columns for powers of one or more of the predictor variables. (Any other nonlinear function is however also legitimate.) This complicates the interpretation of coefficients as slopes, just as though we had done a transformation of a column; see the handout on transformations. Estimation and inference for the coefficients on these predictor variables goes exactly like estimation and inference for any other coefficient.

One column of **x** could be a (nonlinear) function of two or more of the other columns; this is how we represent **interactions**. Usually the interaction column is just a product of two other columns, for a **product** or **multiplicative** interaction; this also complicates the interpretation of coefficients as slopes. (See Ch. 17 on interactions.) Estimation and inference for the coefficients on these predictor variables goes exactly like estimation and inference for any other coefficient.

We can include qualitative predictor variables with k discrete categories or levels by introducing binary indicator variables for $k - 1$ of the levels, and adding them to **x**. The coefficients on these indicators tell us about amounts that are added (or subtracted) to the response for every individual who is a member of that category or level, compared to what would be predicted for an otherwise-identical individual in the baseline category. Equivalently, every category gets its own intercept. Estimation and inference for the coefficients on these predictor variables goes exactly like estimation and inference for any other coefficient.

Interacting the indicator variables for categories with other variables gives coefficients which say what amount is added to the *slope* used for each member of that category (compared to the slope for members of the baseline level). Equivalently, each category gets its own slope. Estimation and inference for the coefficients on these predictor variables goes exactly like estimation and inference for any other coefficient.

Model selection for prediction aims at picking a model which will predict well on new data drawn from the same distribution as the data we've seen. One way to estimate this out-of-sample performance is to look at what the expected squared error would be on new data with the same $\mathbf{x}$ matrix, but a new, independent realization of $\mathbf{y}$. In Ch. 19 on model selection, we showed that

$$\mathbb{E}\left[\frac{1}{n}(\mathbf{Y}'-\widehat{m})^T(\mathbf{Y}'-\widehat{m})\right] \tag{20.38}$$

$$= \mathbb{E}\left[\frac{1}{n}(\mathbf{Y}-\widehat{m})^T(\mathbf{Y}-\widehat{m})\right]+2\frac{1}{n}\sum_{i=1}^{n}\mathrm{Cov}[Y_i,\hat{m}_i]$$

$$= \mathbb{E}\left[\frac{1}{n}(\mathbf{Y}-\widehat{m})^T(\mathbf{Y}-\widehat{m})\right]+\frac{2}{n}\sigma^2\,\mathrm{tr}H \tag{20.39}$$

$$= \mathbb{E}\left[\frac{1}{n}(\mathbf{Y}-\widehat{m})^T(\mathbf{Y}-\widehat{m})\right]+\frac{2}{n}\sigma^2(p+1) \tag{20.40}$$

Mallow's C_p estimates this by

$$MSE+\frac{2}{n}\hat{\sigma}^2(p+1) \tag{20.41}$$

using the $\hat{\sigma}^2$ from the largest, model being selected among (which includes all the other models as special cases).

An alternative is leave-one-out cross-validation, which amounts to

$$\frac{1}{n}\sum_{i=1}^{n}\left(\frac{e_i}{1-H_{ii}}\right)^2 \tag{20.42}$$

C_p and $LOOCV$ converge for large n.

20.3 Gaussian Noise

The Gaussian noise assumption is added on to the other assumptions already made. It is that $\epsilon_i \sim N(0,\sigma^2)$, independent of the predictor variables and all other ϵ_j. Said otherwise, ϵ has a multivariate Gaussian distribution,

$$\epsilon \sim MVN(0,\sigma^2\mathbf{I}) \tag{20.43}$$

Under this assumption, it follows that, since $\widehat{\beta}$ is a linear function of ϵ, it also has a multivariate Gaussian distribution:

$$\widehat{\beta} \sim MVN(\beta,\sigma^2(\mathbf{x}^T\mathbf{x})^{-1}) \tag{20.44}$$

Also,

$$\widehat{\mathbf{m}} \sim MVN(\mathbf{x}\beta,\sigma^2\mathbf{H}) \tag{20.45}$$

It follows from this that

$$\widehat{\beta}_i \sim N(\beta_i, \sigma^2(\mathbf{x}^T\mathbf{x})^{-1}_{i+1,i+1} \tag{20.46}$$

and

$$\widehat{m}_i \sim N(\mathbf{x}_i\beta, \sigma^2 H_{ii}) \tag{20.47}$$

The sampling distribution of the estimated conditional mean at a new point $\mathbf{x}_{new}$ is $N(\mathbf{x}_{new}\beta, \sigma^2\mathbf{x}_{new}(\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T_{new})$.

The mean squared error follows a χ^2-ish distribution:

$$\frac{nMSE}{\sigma^2} \sim \chi^2_{n-p-1} \tag{20.48}$$

Moreover, the MSE is statistically independent of $\widehat{\beta}$. We may therefore define

$$\widehat{\text{se}}\left[\hat{\beta}_i\right] = \hat{\sigma}\sqrt{(\mathbf{x}^T\mathbf{x})^{-1}_{i+1,i+1}} \tag{20.49}$$

and

$$\widehat{\text{se}}\left[\hat{m}_i\right] = \hat{\sigma}\sqrt{H_{ii}} \tag{20.50}$$

and get t distributions:

$$\frac{\hat{\beta}_i - \beta_i}{\widehat{\text{se}}\left[\hat{\beta}_i\right]} \sim t_{n-p-1} \tag{20.51}$$

and

$$\frac{\hat{m}_i - m_i}{\widehat{\text{se}}\left[\hat{m}_i\right]} \sim t_{n-p-1} \tag{20.52}$$

The **Wald test** for the hypothesis that $\beta_i = \beta_i^*$ therefore forms the test statistic

$$\frac{\hat{\beta}_i - \beta_i^*}{\widehat{\text{se}}\left[\hat{\beta}_i\right]} \tag{20.53}$$

and rejects the hypothesis if it is too large (above or below zero) compared to the quantiles of a t_{n-p-1} distribution. The `summary` function of R runs such a test of the hypothesis that $\beta_i = 0$. There is nothing magic or even especially important about testing for a 0 coefficient, and the same test works for testing whether a slope $= 42$ (for example).

Important! The null hypothesis being test is

> Y is a linear function of $X_1, \ldots X_p$, and of no other predictor variables, with independent, constant-variance Gaussian noise, and the coefficient $\beta_i = 0$ exactly.

and the alternative hypothesis is

> Y is a linear function of $X_1, \ldots X_p$, and of no other predictor variables, with independent, constant-variance Gaussian noise, and the coefficient $\beta_i \neq 0$.

The Wald test does not test any of the model assumptions (it presumes them all), and it cannot say whether in an absolutely sense X_i matters for Y; adding or removing other predictors can change whether the true $\beta_i = 0$.

Warning! Retaining the null hypothesis $\beta_i = 0$ can happen if *either* the parameter is precisely estimated, and confidently known to be close to zero, or if it is *im*-precisely estimated, and might as well be zero or something huge on either side. Saying "We can ignore this because we can be quite sure it's small" can make sense; saying "We can ignore this because we have no idea what it is" is preposterous.

To test whether several coefficients are all simultaneously zero, use an F test. The null hypothesis is

> Y is a linear function of $X_1, \ldots X_p$, and of no other predictor variables, with independent, constant-variance Gaussian noise, and the coefficients for all but q of the variables are exactly 0, $\beta_{q+1} = \beta_{q+2} \cdots \beta_p = 0$.

(You can always re-order the predictor variables so that you're saying coefficients 1 through q are free, and $q+1$ through p are fixed to zero.) The alternative hypothesis is

> Y is a linear function of $X_1, \ldots X_p$, and of no other predictor variables, with independent, constant-variance Gaussian noise, and the coefficient $\beta_{q+1}, \beta_{q+2}, \ldots \beta_q \neq 0$.

The smaller, null model leads to an estimate of σ^2, $\hat{\sigma}^2_{null}$, and the larger, alternative model gives us the estimate $\hat{\sigma}^2_{full}$. The F statistic is

$$F_{stat} = \frac{(\hat{\sigma}^2_{null} - \hat{\sigma}^2_{full})/(p-q)}{\hat{\sigma}^2_{full}/(n-p-1)} \tag{20.54}$$

The numerator is basically about how much the MSE shrinks for each extra parameter we throw at fitting the data; the denominator is an estimate of how big an improvement-per-degree-of-freedom we should see under the null hypothesis. Under that null hypothesis,

$$F_{stat} \sim F_{p-q,n-p-1} \tag{20.55}$$

Note that the F test when $q = p-1$ becomes identical to a t-test.

If $0 < q < p$, we have a "partial" F test. A "full" F test sets $q = 0$, i.e., it tests the null hypothesis of an intercept-only model (with independent, constant-variance Gaussian noise) against the alternative of the linear model on $X_1, \ldots X_p$ (and only those variables, with independent, constant-variance Gaussian noise). This is only of interest under very unusual circumstances.

Once again, no F test is capable of checking any modeling assumptions. This is because both the null hypothesis and the alternative hypothesis presume that the all of the modeling assumptions are exactly correct.

Inverting the Wald test yields confidence intervals for each coefficient: a $1-\alpha$ confidence interval for β_i is

$$\hat{\beta}_i \pm \widehat{\text{se}}[\beta_i]\, t_{n-p-1}(\alpha/2) \tag{20.56}$$

Inverting the F test leads to ellipse-shaped (in more than two dimensions, "ellipsoidal") confidence regions. (See Chapter 16 for the detailed formula, which involves matrix inversion.) These make a *simultaneous* guarantee: either *all* the parameters are inside the confidence region, or we were very unlucky when we got our data. A simpler way to get a simultaneous confidence region for all p parameters is to use $1-\alpha/p$ confidence intervals for each one ("Bonferroni correction").

Turning to the fitted values, by entirely parallel reasoning, we may test any hypothesis of the form $m_i = m_i^*$ by a t-test. More practically, a $1-\alpha$ confidence interval for the fitted value (conditional expectation) is

$$\hat{m}_i \pm \widehat{\text{se}}[\hat{m}_i]\, t_{n-p-1}(\alpha/2) \tag{20.57}$$

The standardized residuals do not quite have a t distribution, because $\hat{\sigma}$ is a function of all the e_is, hence the numerator in the definition of r_i is not independent of the denominator. But the cross-validated or studentized residuals are:

1. Temporarily hold out data point i
2. Re-estimate the coefficients to get $\widehat{\beta}^{(-i)}$ and $\hat{\sigma}^{(-i)}$
3. Make a prediction for y_i, $\hat{m}_i^{(-i)}$
4. Calculate

$$t_i = \frac{y_i - \hat{m}_i^{(-i)}}{\hat{\sigma}^{(-i)} + \widehat{\text{se}}\left[\hat{m}_i^{(-i)}\right]} \tag{20.58}$$

This can be done without recourse to actually re-fitting the model[2]:

$$t_i = r_i \sqrt{\frac{n-p-1}{n-p-r_i^2}} \tag{20.59}$$

(Note that for large n, this is typically extremely close to r_i.) Once we have it, there is a sampling distribution[3] for t_i:

$$t_i \sim t_{n-p-2} \tag{20.60}$$

[2]Like the definition of cross-validated or studentized residuals, this next formula does not need the Gaussian noise assumption.

[3]Under the Gaussian noise assumption.

(The -2 is because we're using $n-1$ data points to estimate $p+1$ coefficients.)

Cook's distance for point i is the sum of the (squared) changes to all the fitted values if i was omitted; it is[4]

$$D_i = \frac{1}{p+1} e_i^2 \frac{H_{ii}}{(1-H_{ii})^2} \tag{20.61}$$

There is a tradition of being worried if $D_i \geq F_{p,n-p-2}(0.1)$, but no hard proofs behind it.

Under the Gaussian assumption, one can calculate a likelihood for the data. The log-likelihood is, up to constants independent of the parameters,

$$L = -\frac{n}{2} \log MSE \tag{20.62}$$

since the MSE is the maximum likelihood estimate of σ^2 (even though it is slightly biased). The likelihood ratio test which can be used in place of the F test has as its test statistic

$$\begin{aligned}
\Lambda &\equiv \log \frac{L_{full}}{L_{null}} && (20.63)\\
&= \log L_{full} - \log L_{null} && (20.64)\\
&= \frac{n}{2}\left(-\log MSE_{full} + \log MSE_{null}\right) && (20.65)\\
&= \frac{n}{2} \log \frac{MSE_{null}}{MSE_{full}} && (20.66)
\end{aligned}$$

Under the null hypothesis, $2\Lambda \sim \chi^2_{p-q}$ for large n.

The log-likelihood can also be used to calculate a model selection criterion, the Akaike information criterion (AIC):

$$AIC = L - (p+1) \tag{20.67}$$

Sometimes this is written with a factor of 2, or of -1 (so that smaller values of AIC are preferred), or of -2 (ditto). As $n \rightarrow \infty$, the model selected by AIC becomes identical to the model selected by C_p or LOOCV. (In fact, the AIC score of the true model converges on its C_p or LOOCV score.) AIC is best thought of as a very fast (because very simple) approximation to leave-one-out CV.

[4]Again, no Gaussian noise assumption is needed for the next formula.

Chapter 21

Weighted and Generalized Least Squares

21.1 Weighted Least Squares

When we use ordinary least squares to estimate linear regression, we (naturally) minimize the mean squared error:

$$MSE(\mathbf{b}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \mathbf{x}_{i\cdot}\beta)^2 \tag{21.1}$$

The solution is of course

$$\widehat{\beta}_{OLS} = (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{y} \tag{21.2}$$

We could instead minimize the *weighted* mean squared error,

$$WMSE(\mathbf{b}, w_1, \ldots w_n) = \frac{1}{n}\sum_{i=1}^{n} w_i(y_i - \mathbf{x}_{i\cdot}\mathbf{b})^2 \tag{21.3}$$

This includes ordinary least squares as the special case where all the weights $w_i = 1$. We can solve it by the same kind of linear algebra we used to solve the ordinary linear least squares problem. If we write $\mathbf{w}$ for the matrix with the w_i on the diagonal and zeroes everywhere else, then

$$\begin{aligned} WMSE &= n^{-1}(\mathbf{y}-\mathbf{x}\mathbf{b})^T\mathbf{w}(\mathbf{y}-\mathbf{x}\mathbf{b}) & (21.4)\\ &= \frac{1}{n}\left(\mathbf{y}^T\mathbf{w}\mathbf{y} - \mathbf{y}^T\mathbf{w}\mathbf{x}\mathbf{b} - \mathbf{b}^T\mathbf{x}^T\mathbf{w}\mathbf{y} + \mathbf{b}^T\mathbf{x}^T\mathbf{w}\mathbf{x}\mathbf{b}\right) & (21.5)\end{aligned}$$

Differentiating with respect to $\mathbf{b}$, we get as the gradient

$$\nabla_{\mathbf{b}} WMSE = \frac{2}{n}\left(-\mathbf{x}^T\mathbf{w}\mathbf{y} + \mathbf{x}^T\mathbf{w}\mathbf{x}\mathbf{b}\right)$$

Setting this to zero at the optimum and solving,

$$\widehat{\beta}_{WLS} = (\mathbf{x}^T \mathbf{w} \mathbf{x})^{-1} \mathbf{x}^T \mathbf{w} \mathbf{y} \tag{21.6}$$

But why would we want to minimize Eq. 21.3?

1. *Focusing accuracy.* We may care very strongly about predicting the response for certain values of the input — ones we expect to see often again, ones where mistakes are especially costly or embarrassing or painful, etc. — than others. If we give the points near that region big weights, and points elsewhere smaller weights, the regression will be pulled towards matching the data in that region.

2. *Discounting imprecision.* Ordinary least squares minimizes the squared error when the variance of the noise terms ϵ is constant over all observations, so we're measuring the regression function with the same precision elsewhere. This situation, of constant noise variance, is called **homoskedasticity**. Often however the magnitude of the noise is not constant, and the data are **heteroskedastic**.

   When we have heteroskedasticity, ordinary least squares is no longer the optimal estimate — we'll see presently that other estimators can be unbiased and have smaller variance. If however we know the noise variance σ_i^2 at each measurement i, and set $w_i = 1/\sigma_i^2$, we get minimize the variance of estimation.

   To say the same thing slightly differently, there's just no way that we can estimate the regression function as accurately where the noise is large as we can where the noise is small. Trying to give equal attention to all values of X is a waste of time; we should be more concerned about fitting well where the noise is small, and expect to fit poorly where the noise is big.

3. *Sampling bias.* In many situations, our data comes from a survey, and some members of the population may be more likely to be included in the sample than others. When this happens, the sample is a biased representation of the population. If we want to draw inferences about the population, it can help to give more weight to the kinds of data points which we've under-sampled, and less to those which were over-sampled. In fact, typically the weight put on data point i would be inversely proportional to the probability of i being included in the sample (exercise 1). Strictly speaking, if we are willing to believe that linear model is exactly correct, that there are no omitted variables, and that the inclusion probabilities p_i do not vary with y_i, then this sort of survey weighting is redundant (DuMouchel and Duncan, 1983). When those assumptions are not met — when there're non-linearities, omitted variables, or "selection on the dependent variable" — survey weighting is advisable, if we know the inclusion probabilities fairly well.

   The same trick works under the same conditions when we deal with "covariate shift", a change in the distribution of X. If the old probability density function was $p(x)$ and the new one is $q(x)$, the weight we'd want to use is $w_i = q(x_i)/p(x_i)$ (Quiñonero-Candela *et al.*, 2009). This can involve estimating both densities, or their ratio (topics we'll cover in 402).

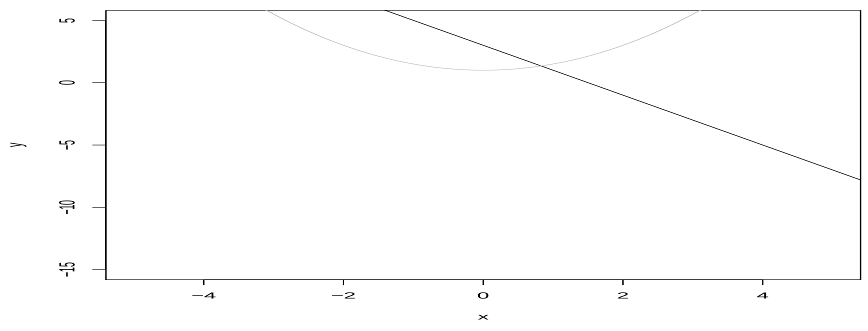


FIGURE 21.1: *Black line: Linear response function ($y = 3 - 2x$). Grey curve: standard deviation as a function of x ($\sigma(x) = 1 + x^2/2$). (Code deliberately omitted; can you reproduce this figure?)*

4. *Doing something else.* There are a number of other optimization problems which can be transformed into, or approximated by, weighted least squares. The most important of these arises from **generalized linear models**, where the mean response is some nonlinear function of a linear predictor; we will look at them in 402.

In the first case, we decide on the weights to reflect our priorities. In the third case, the weights come from the optimization problem we'd really rather be solving. What about the second case, of heteroskedasticity?

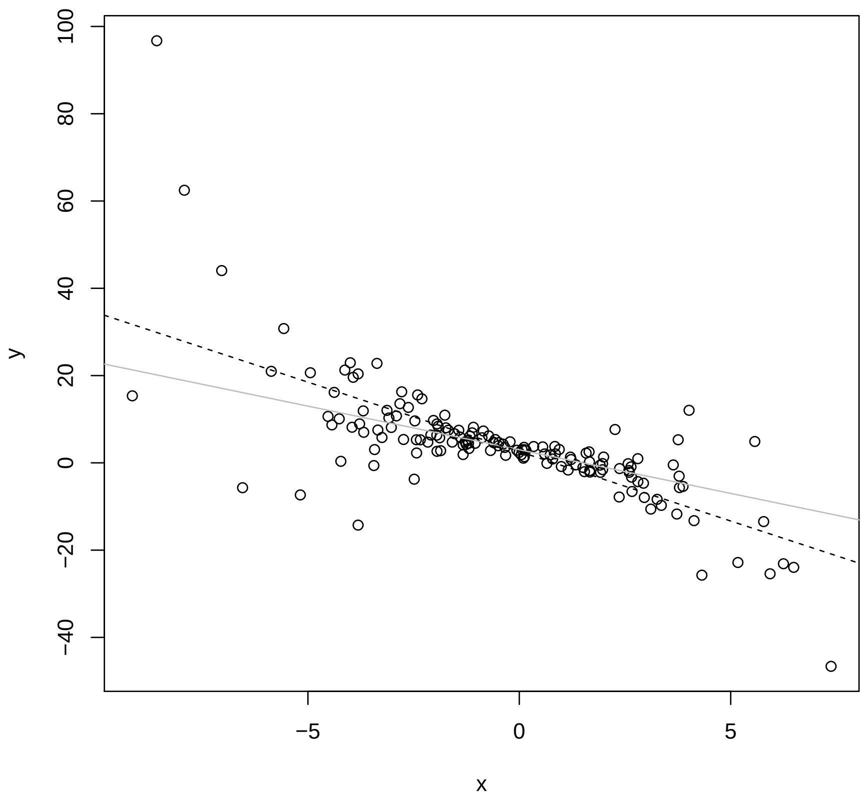
21.2 Heteroskedasticity

Suppose the noise variance is itself variable. For example, Figure 21.1 shows a simple linear relationship between the predictors X and the response Y, but also a nonlinear relationship between X and $\mathrm{Var}[Y]$.

In this particular case, the ordinary least squares estimate of the regression line is $2.59 - 3.18x$, with R reporting standard errors in the coefficients of ± 0.79 and 0.25, respectively. Those are however calculated under the assumption that the noise is homoskedastic, which it isn't. And in fact we can see, pretty much, that there is heteroskedasticity — if looking at the scatter-plot didn't convince us, we could always plot the residuals against x, which we should do anyway.

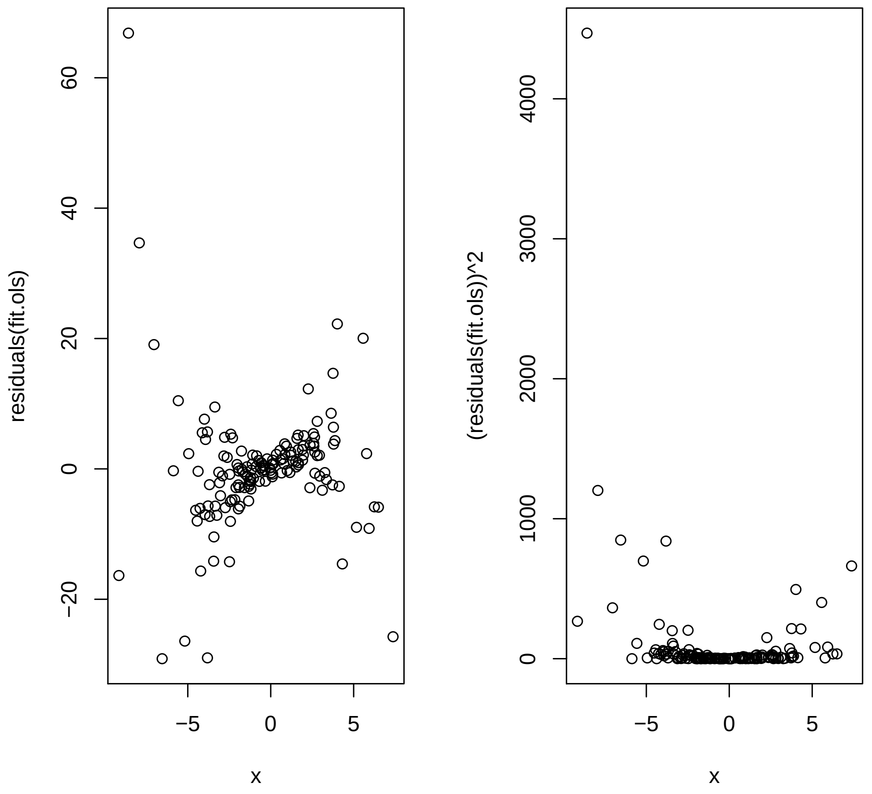
To see whether that makes a difference, let's re-do this many times with different draws from the same model (Figure 21.4).

Running `ols.heterosked.error.stats(100)` produces 10^4 random samples which all have the same x values as the first one, but different values of y, generated however from the same model. It then uses those samples to get the standard error of the ordinary least squares estimates. (Bias remains a non-issue.) What we find is the standard error of the intercept is only a little inflated (simulation value of 0.69 versus official value of 0.79), but the standard error of the slope is much larger than what R reports,



```
# Plot the data
plot(x, y)
# Plot the true regression line
abline(a = 3, b = -2, col = "grey")
# Fit by ordinary least squares
fit.ols = lm(y ~ x)
# Plot that line
abline(fit.ols, lty = "dashed")
```

FIGURE 21.2: *Scatter-plot of $n = 150$ data points from the above model. (Here X is Gaussian with mean 0 and variance 9.) Grey: True regression line. Dashed: ordinary least squares regression line.*



```
par(mfrow = c(1, 2))
plot(x, residuals(fit.ols))
plot(x, (residuals(fit.ols))^2)
par(mfrow = c(1, 1))
```

FIGURE 21.3: *Residuals (left) and squared residuals (right) of the ordinary least squares regression as a function of x. Note the much greater range of the residuals at large absolute values of x than towards the center; this changing dispersion is a sign of heteroskedasticity.*

```
# Generate more random samples from the same model and the same x values,
# but different y values Inputs: number of samples to generate Presumes: x
# exists and is defined outside this function Outputs: errors in linear
# regression estimates
ols.heterosked.example = function(n) {
    y = 3 - 2 * x + rnorm(n, 0, sapply(x, function(x) {
        1 + 0.5 * x^2
    }))
    fit.ols = lm(y ~ x)
    # Return the errors
    return(fit.ols$coefficients - c(3, -2))
}

# Calculate average-case errors in linear regression estimates (SD of slope
# and intercept) Inputs: number of samples per replication, number of
# replications (defaults to 10,000) Calls: ols.heterosked.example Outputs:
# standard deviation of intercept and slope
ols.heterosked.error.stats = function(n, m = 10000) {
    ols.errors.raw = t(replicate(m, ols.heterosked.example(n)))
    # transpose gives us a matrix with named columns
    intercept.sd = sd(ols.errors.raw[, "(Intercept)"])
    slope.sd = sd(ols.errors.raw[, "x"])
    return(list(intercept.sd = intercept.sd, slope.sd = slope.sd))
}
```

FIGURE 21.4: *Functions to generate heteroskedastic data and fit OLS regression to it, and to collect error statistics on the results.*

0.5 versus 0.25. Since the intercept is fixed by the need to make the regression line go through the center of the data, the real issue here is that our estimate of the slope is much less precise than ordinary least squares makes it out to be. Our estimate is still consistent, but not as good as it was when things were homoskedastic. Can we get back some of that efficiency?

FIGURE 21.5: *Statistician (right) consulting the Oracle of Regression (left) about the proper weights to use to overcome heteroskedasticity.* *(Image from* `http://en.wikipedia.org/wiki/Image:Pythia1.jpg`*.)*

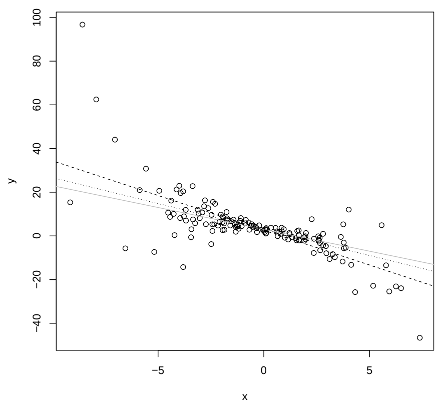
21.2.1 Weighted Least Squares as a Solution to Heteroskedasticity

Suppose we visit the Oracle of Regression (Figure 21.5), who tells us that the noise has a standard deviation that goes as $1+x^2/2$. We can then use this to improve our regression, by solving the weighted least squares problem rather than ordinary least squares (Figure 21.6).

The estimated line is now $2.92 - 2.38x$, with reported standard errors of 0.34 and 0.19. Does this check out with simulation? (Figure 21.7.)

Unsurprisingly, yes. The standard errors from the simulation are 0.34 for the intercept and 0.19 for the slope, so R's internal calculations are working very well.

Why does putting these weights into WLS improve things?



```
# Plot the data
plot(x, y)
# Plot the true regression line
abline(a = 3, b = -2, col = "grey")
# Fit by ordinary least squares
fit.ols = lm(y ~ x)
# Plot that line
abline(fit.ols, lty = "dashed")
fit.wls = lm(y ~ x, weights = 1/(1 + 0.5 * x^2))
abline(fit.wls, lty = "dotted")
```

FIGURE 21.6: *Figure 21.2, plus the weighted least squares regression line (dotted).*

```
### As previous two functions, but with weighted regression

# Generate random sample from model (with fixed x), fit by weighted least
# squares Inputs: number of samples Presumes: x fixed outside function
# Outputs: errors in parameter estimates
wls.heterosked.example = function(n) {
    y = 3 - 2 * x + rnorm(n, 0, sapply(x, function(x) {
        1 + 0.5 * x^2
    }))
    fit.wls = lm(y ~ x, weights = 1/(1 + 0.5 * x^2))
    # Return the errors
    return(fit.wls$coefficients - c(3, -2))
}

# Calculate standard errors in parameter estiamtes over many replications
# Inputs: number of samples per replication, number of replications
# (defaults to 10,000) Calls: wls.heterosked.example Outputs: standard
# deviation of estimated intercept and slope
wls.heterosked.error.stats = function(n, m = 10000) {
    wls.errors.raw = t(replicate(m, wls.heterosked.example(n)))
    # transpose gives us a matrix with named columns
    intercept.sd = sd(wls.errors.raw[, "(Intercept)"])
    slope.sd = sd(wls.errors.raw[, "x"])
    return(list(intercept.sd = intercept.sd, slope.sd = slope.sd))
}
```

FIGURE 21.7: *Linear regression of heteroskedastic data, using weighted least-squared regression.*

21.2.2 Some Explanations for Weighted Least Squares

Qualitatively, the reason WLS with inverse variance weights works is the following. OLS cares equally about the error at each data point.[1] Weighted least squares, naturally enough, tries harder to match observations where the weights are big, and less hard to match them where the weights are small. But each y_i contains not only the true regression function $m(x_i)$ but also some noise ϵ_i. The noise terms have large magnitudes where the variance is large. So we should want to have small weights where the noise variance is large, because there the data tends to be far from the true regression. Conversely, we should put big weights where the noise variance is small, and the data points are close to the true regression.

The qualitative reasoning in the last paragraph doesn't explain why the weights should be inversely proportional to the variances, $w_i \propto 1/\sigma_i^2$ — why not $w_i \propto 1/\sigma_i$, for instance? Look at the equation for the WLS estimates again:

$$\widehat{\beta}_{WLS} = (\mathbf{x}^T\mathbf{w}\mathbf{x})^{-1}\mathbf{x}^T\mathbf{w}\mathbf{y} \tag{21.7}$$

Imagine holding $\mathbf{x}$ constant, but repeating the experiment multiple times, so that we get noisy values of $\mathbf{y}$. In each experiment, $Y_i = \mathbf{x}_{i\cdot}\beta + \epsilon_i$, where $\mathbb{E}[\epsilon_i|\mathbf{x}] = 0$ and $\mathrm{Var}[\epsilon_i|\mathbf{x}] = \sigma_i^2$. So

$$\begin{aligned} \widehat{\beta}_{WLS} &= (\mathbf{x}^T\mathbf{w}\mathbf{x})^{-1}\mathbf{x}^T\mathbf{w}\mathbf{x}\beta + (\mathbf{x}^T\mathbf{w}\mathbf{x})^{-1}\mathbf{x}^T\mathbf{w}\epsilon & (21.8)\\ &= \beta + (\mathbf{x}^T\mathbf{w}\mathbf{x})^{-1}\mathbf{x}^T\mathbf{w}\epsilon & (21.9)\end{aligned}$$

Since $\mathbb{E}[\epsilon|\mathbf{x}] = 0$, the WLS estimator is unbiased:

$$\mathbb{E}\left[\widehat{\beta}_{WLS}|\mathbf{x}\right] = \beta \tag{21.10}$$

In fact, for the j^{th} coefficient,

$$\begin{aligned} \widehat{\beta}_j &= \beta_j + [(\mathbf{x}^T\mathbf{w}\mathbf{x})^{-1}\mathbf{x}^T\mathbf{w}\epsilon]_j & (21.11)\\ &= \beta_j + \sum_{i=1}^{n} k_{ji}(w)\epsilon_i & (21.12)\end{aligned}$$

where in the last line I have bundled up $(\mathbf{x}^T\mathbf{w}\mathbf{x})^{-1}\mathbf{x}^T\mathbf{w}$ as a matrix $\mathbf{k}(w)$, with the argument to remind us that it depends on the weights. Since the WLS estimate is unbiased, it's natural to want it to also have a small variance, and

$$\mathrm{Var}\left[\widehat{\beta}_j\right] = \sum_{i=1}^{n} k_{ji}(w)\sigma_i^2 \tag{21.13}$$

It can be shown — the result is called the **generalized Gauss-Markov theorem** — that picking weights to minimize the variance in the WLS estimate has the unique solution

[1] Less anthropomorphically, the objective function in Eq. 21.1 has the same derivative with respect to the squared error at each point, $\frac{\partial MSE}{\partial e_i^2} = \frac{1}{n}$.

$w_i = 1/\sigma_i^2$. It does not require us to assume the noise is Gaussian, but the proof does need a few tricks (see §21.3).

A less general but easier-to-grasp result comes from adding the assumption that the noise around the regression line is Gaussian — that

$$Y = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p + \epsilon,\ \epsilon \sim \mathcal{N}(0, \sigma_x^2) \tag{21.14}$$

The log-likelihood is then (Exercise 2)

$$-\frac{n}{2}\ln 2\pi - \frac{1}{2}\sum_{i=1}^{n}\log\sigma_i^2 - \frac{1}{2}\sum_{i=1}^{n}\frac{(y_i - \mathbf{x}_{i\cdot}\mathbf{b})^2}{\sigma_i^2} \tag{21.15}$$

If we maximize this with respect to β, everything except the final sum is irrelevant, and so we minimize

$$\sum_{i=1}^{n}\frac{(y_i - \mathbf{x}_{i\cdot}\mathbf{b})^2}{\sigma_i^2} \tag{21.16}$$

which is just weighted least squares with $w_i = 1/\sigma_i^2$. So, if the probabilistic assumption holds, WLS is the efficient maximum likelihood estimator.

21.3 The Gauss-Markov Theorem

We've seen that when we do weighted least squares, our estimates of β are linear in $\mathbf{Y}$, and unbiased (Eq. 21.10):

$$\widehat{\beta_{WLS}} = (\mathbf{x}^T\mathbf{w}\mathbf{x})^{-1}\mathbf{x}^T\mathbf{w}\mathbf{y} \tag{21.17}$$

$$\mathbb{E}\left[\widehat{\beta_{WLS}}\right] = \beta \tag{21.18}$$

What we'd like to show is that using the weights $w_i = 1/\sigma_i^2$ is somehow optimal. Like any optimality result, it is crucial to lay out carefully the range of possible alternatives, and the criterion by which those alternatives will be compared. The classical optimality result for estimating linear models is the **Gauss-Markov theorem**, which takes the range of possibilities to be *linear, unbiased estimators of* β, and the criterion to be *variance of the estimator*. I will return to both these choices at the end of this section.

Any linear estimator, say $\widetilde{\beta}$, could be written as

$$\widetilde{\beta} = \mathbf{q}\mathbf{y}$$

where $\mathbf{q}$ would be a $(p+1)\times n$ matrix, in general a function of $\mathbf{x}$, weights, the phase of the moon, etc. (For OLS, $\mathbf{q} = (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T$.) For $\widetilde{\beta}$ to be an unbiased estimator, we must have

$$\mathbb{E}[\mathbf{q}\mathbf{Y}|\mathbf{x}] = \mathbf{q}\mathbf{x}\beta = \beta$$

Since this must hold for all β and all $\mathbf{x}$, we have to have $\mathbf{q}\mathbf{x} = \mathbf{I}$.[2] (Sanity check: this works for OLS.) The variance is then

$$\text{Var}\left[\mathbf{q}\mathbf{Y}|\mathbf{x}\right] = \mathbf{q}\text{Var}\left[\epsilon|\mathbf{x}\right]\mathbf{q}^T = \mathbf{q}\Sigma\mathbf{q} \tag{21.19}$$

where I abbreviate the mouthful $\text{Var}\left[\epsilon|\mathbf{x}\right]$ by Σ. We could then try to differentiate this with respect to $\mathbf{q}$, set the derivative to zero, and solve, but this gets rather messy, since in addition to the complications of matrix calculus, we'd need to enforce the unbiasedness constraint $\mathbf{q}\mathbf{x} = \mathbf{I}$ somehow.

Instead of the direct approach, we'll use a classic piece of trickery. Set

$$\mathbf{k} \equiv (\mathbf{x}^T\Sigma^{-1}\mathbf{x})^{-1}\mathbf{x}^T\Sigma^{-1}$$

which is the estimating matrix for weighted least squares. Now, whatever $\mathbf{q}$ might be, we can always write

$$\mathbf{q} = \mathbf{k} + \mathbf{r} \tag{21.20}$$

for some matrix $\mathbf{r}$. The unbiasedness constraint on $\mathbf{q}$ translates into

$$\mathbf{r}\mathbf{x} = 0$$

because $\mathbf{k}\mathbf{x} = \mathbf{I}$. Now we substitute Eq. 21.20 into Eq. 21.19:

$$\begin{aligned}
\text{Var}\left[\tilde{\beta}\right] &= (\mathbf{k}+\mathbf{r})\Sigma(\mathbf{k}+\mathbf{r})^T & (21.21)\\
&= (\mathbf{k}+\mathbf{r})\Sigma^{-1}(\mathbf{k}+\mathbf{r})^T & (21.22)\\
&= \mathbf{k}\Sigma\mathbf{k}^T + \mathbf{r}\Sigma\mathbf{k}^T + \mathbf{k}\Sigma\mathbf{r}^T + \mathbf{r}\Sigma\mathbf{r}^T & (21.23)\\
&= (\mathbf{x}^T\Sigma^{-1}\mathbf{x})^{-1}\mathbf{x}^T\Sigma^{-1}\Sigma\Sigma^{-1}\mathbf{x}(\mathbf{x}^T\Sigma^{-1}\mathbf{x})^{-1} & (21.24)\\
&\quad +\mathbf{r}\Sigma\Sigma^{-1}\mathbf{x}(\mathbf{x}^T\Sigma^{-1}\mathbf{x})^{-1} & \\
&\quad +(\mathbf{x}^T\Sigma^{-1}\mathbf{x})^{-1}\mathbf{x}^T\Sigma^{-1}\Sigma\mathbf{r}^T & \\
&\quad +\mathbf{r}\Sigma\mathbf{r}^T & \\
&= (\mathbf{x}^T\Sigma^{-1}\mathbf{x})^{-1}\mathbf{x}^T\Sigma^{-1}\mathbf{x}(\mathbf{x}^T\Sigma^{-1}\mathbf{x})^{-1} & (21.25)\\
&\quad +\mathbf{r}\mathbf{x}(\mathbf{x}^T\Sigma^{-1}\mathbf{x})^{-1} + (\mathbf{x}^T\Sigma^{-1}\mathbf{x})^{-1}\mathbf{x}^T\mathbf{r}^T & \\
&\quad +\mathbf{r}\Sigma\mathbf{r}^T & \\
&= (\mathbf{x}^T\Sigma^{-1}\mathbf{x})^{-1} + \mathbf{r}\Sigma\mathbf{r}^T & (21.26)
\end{aligned}$$

where the last step uses the fact that $\mathbf{r}\mathbf{x} = 0$ (and so $\mathbf{x}^T\mathbf{r}^T = 0^T$).

Since Σ is a covariance matrix, it's positive definite, meaning that $a\Sigma a^T \geq 0$ for any vector a. This applies in particular to the vector $\mathbf{r}_{i\cdot}$, i.e., the i^{th} row of $\mathbf{r}$. But

$$\text{Var}\left[\tilde{\beta}_i\right] = (\mathbf{x}^T\Sigma^{-1}\mathbf{x})^{-1}_{ii} + \mathbf{r}_{i\cdot}\mathbf{w}_0{}^{-1}\mathbf{r}_{i\cdot}^T$$

which must therefore be strictly larger than $(\mathbf{x}^T\Sigma^{-1}\mathbf{x})^{-1}_{ii}$, the variance we'd get from using weighted least squares.

[2]This doesn't mean that $\mathbf{q} = \mathbf{x}^{-1}$; x doesn't have an inverse!

FIGURE 21.8: *The Oracle may be out (left), or too creepy to go visit (right). What then? (Left, the sacred oak of the Oracle of Dodona, copyright 2006 by Flickr user "essayen",* `http://flickr.com/photos/essayen/245236125/`*; right, the entrace to the cave of the Sibyl of Cumæ, copyright 2005 by Flickr user "pverdicchio",* `http://flickr.com/photos/occhio/17923096/`*. Both used under Creative Commons license.)*

We conclude that WLS, with the weight matrix $\mathbf{w}$ equal to the inverse variance matrix Σ^{-1}, the least variance among all possible linear, unbiased estimators of the regression coefficients.

Notes:

1. If all the noise variances are equal, then we've proved the optimality of OLS.
2. The theorem doesn't rule out linear, biased estimators with smaller variance. As an example, albeit a trivial one, $\mathbf{0}\mathbf{y}$ is linear and has variance $\mathbf{0}$, but is (generally) very biased.
3. The theorem also doesn't rule out non-linear unbiased estimators of smaller variance. Or indeed non-linear biased estimators of even smaller variance.
4. The proof actually doesn't require the variance matrix to be diagonal.

21.4 Finding the Variance and Weights

All of this was possible because the Oracle told us what the variance function was. What do we do when the Oracle is not available (Figure 21.8)?

Sometimes we can work things out for ourselves, without needing an oracle.

- We know, empirically, the precision of our measurement of the response variable — we know how precise our instruments are, or the response is really an average of several measurements so we can use their standard deviations, etc.
- We know how the noise in the response must depend on the input variables. For example, when taking polls or surveys, the variance of the proportions we find should be inversely proportional to the sample size. So we can make the weights proportional to the sample size.

Both of these outs rely on kinds of background knowledge which are easier to get in the natural or even the social sciences than in many industrial applications. However, there are approaches for other situations which try to use the observed residuals to get estimates of the heteroskedasticity; this is the topic of the next section.

21.4.1 Variance Based on Probability Considerations

There are a number of situations where we can reasonably base judgments of variance, or measurement variance, on elementary probability.

Multiple measurements The easiest case is when our measurements of the response are actually averages over individual measurements, each with some variance σ^2. If some Y_i are based on averaging more individual measurements than others, there will be heteroskedasticity. The variance of the average of n_i uncorrelated measurements will be σ^2/n_i, so in this situation we could take $w_i \propto n_i$.

Binomial counts Suppose our response variable is a count, derived from a binomial distribution, i.e., $Y_i \sim \mathrm{Binom}(n_i, p_i)$. We would usually model p_i as a function of the predictor variables — at this level of statistical knowledge, a linear function. This would imply that Y_i had expectation $n_i p_i$, and variance $n_i p_i(1-p_i)$. We would be well-advised to use this formula for the variance, rather than pretending that all observations had equal variance.

Proportions based on binomials If our response variable is a proportion based on a binomial, we'd see an expectation value of p_i and a variance of $\frac{p_i(1-p_i)}{n_i}$. Again, this is not equal across different values of n_i, or for that matter different values of p_i.

Poisson counts Binomial counts have a hard upper limit, n_i; if the upper limit is immense or even (theoretically) infinite, we may be better off using a Poisson distribution. In such situations, the mean of the Poisson λ_i will be a (possibly-linear) function of the predictors, and the variance will *also* be equal to λ_i.

Other counts The binomial and Poisson distributions rest on independence across "trials" (whatever those might be). There are a range of discrete probability models which allow for correlation across trials (leadings to more or less variance). These may, in particular situations, be more appropriate.

21.4.1.1 Example: The Economic Mobility Data

The data set on economic mobility we've used in a number of assignments and examples actually contains a bunch of other variables in addition to the covariates we've looked at (short commuting times and latitude and longitude). While reserving the full data set for later use, let's look one of the additional covariates, namely population.

To see why this might be relevant, recall that our response variable is the *fraction* of children who, in each community, were born into the lowest 20% of the income distribution during 1980–1982 and nonetheless make it into the top 20% by age 30; we're looking at a proportion. Different communities will have had different numbers of children born in the relevant period, generally proportional to their total population. Treating the observed fraction for New York City as being just as far from its *expected* rate of mobility as that for Piffleburg, WI is asking for trouble.

Once we have population, there is a very notable pattern: the most extreme levels of mobility are all for very small communities (Figure 21.9).

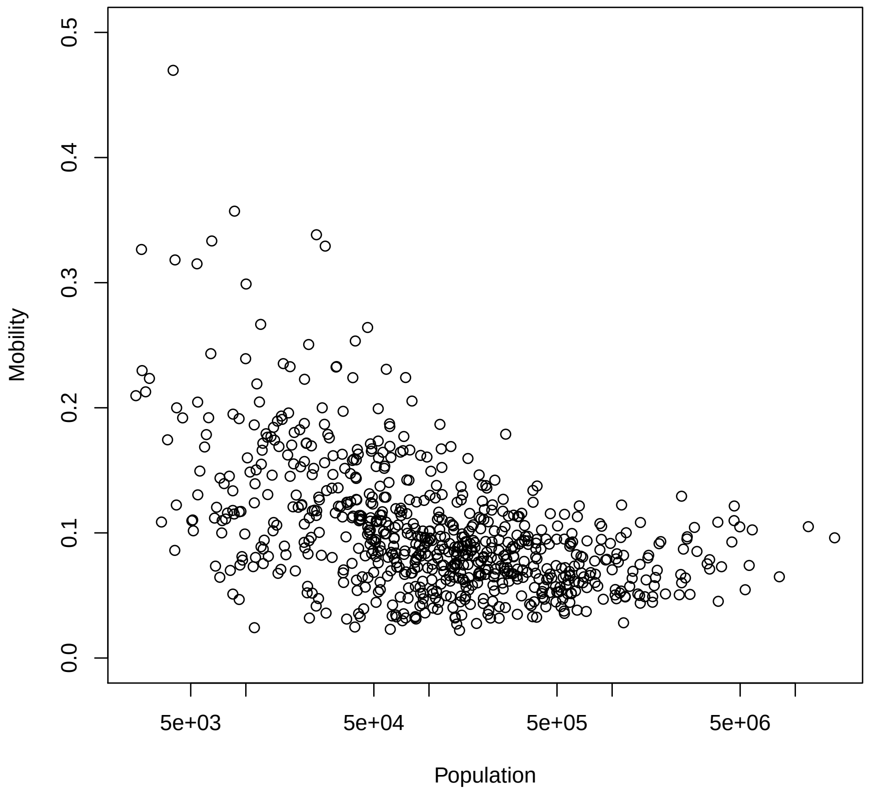
While we do not know the exact number of children for each community, it is not unreasonable to take that as proportional to the total population. The binomial standard error in the observed fraction will therefore be $\propto \sqrt{\frac{p_i(1-p_i)}{n_i}}$.

```
mobility$MobSE <- with(mobility, sqrt(Mobility * (1 - Mobility)/Population))
```

Let us now plot the rate of economic mobility against the fraction of workers with short commutes, and decorate it with error bars reflecting these standard errors (Figure 21.10).

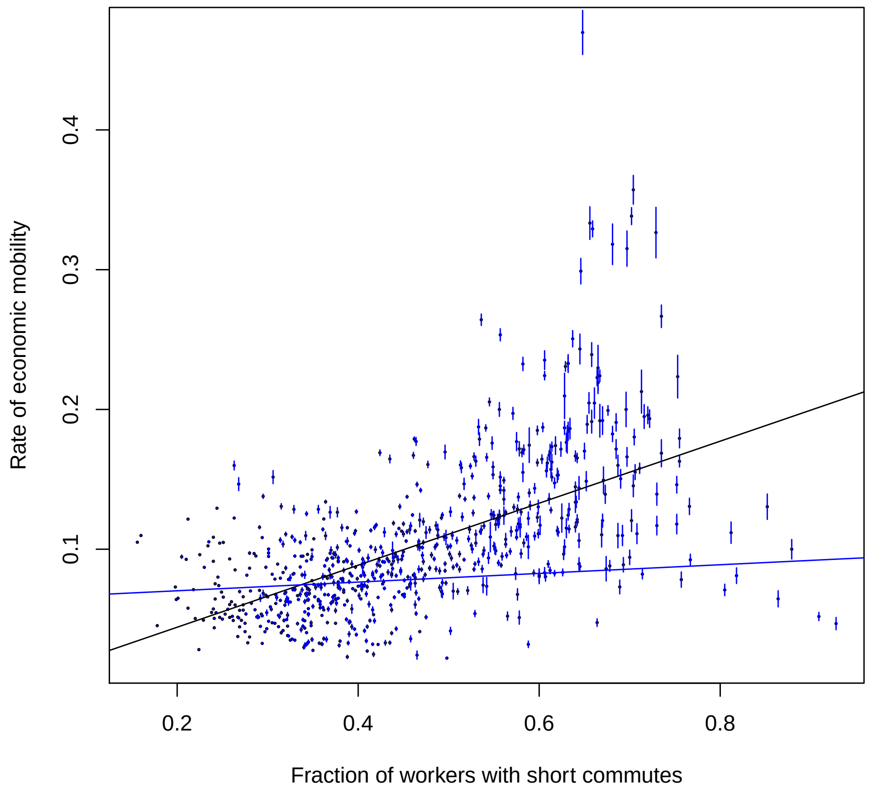
Now, there are reasons why this is not necessarily the last word on using weighted least squares here. One is that if we actually believed our model, we should be using the *predicted* mobility as the p_i in $\sqrt{\frac{p_i(1-p_i)}{n_i}}$, rather than the observed mobility. Another is that the binomial model assumes *independence* across "trials" (here, children). But, by definition, at most, and at least, 20% of the population ends up in the top 20% of the income distribution[3]. It's fairly clear, however, that simply *ignoring* differences in the sizes of communities is unwise.

[3]Cf. Gore Vidal: "It is not enough to succeed; others must also fail."



```
mobility <- read.csv("http://www.stat.cmu.edu/~cshalizi/mreg/15/lectures/24--25/mobility2.csv")
plot(Mobility ~ Population, data = mobility, log = "x", ylim = c(0, 0.5))
```

FIGURE 21.9: *Rate of economic mobility plotted against population, with a logarithmic scale on the latter, horizontal axis. Notice decreasing spread at larger population.*



```
plot(Mobility ~ Commute, data = mobility, xlab = "Fraction of workers with short commutes",
    ylab = "Rate of economic mobility", pch = 19, cex = 0.2)
with(mobility, segments(x0 = Commute, y0 = Mobility + 2 * MobSE, x1 = Commute,
    y1 = Mobility - 2 * MobSE, col = "blue"))
mob.lm <- lm(Mobility ~ Commute, data = mobility)
mob.wlm <- lm(Mobility ~ Commute, data = mobility, weight = 1/MobSE^2)
abline(mob.lm)
abline(mob.wlm, col = "blue")
```

FIGURE 21.10: *Mobility versus the fraction of workers with short commute, with ± 2 standard deviation error bars (vertical blue bars), and the OLS linear fit (black line) and weighted least squares (blue line). Note that the error bars for some larger communities are smaller than the diameter of the dots.*

21.5 Conditional Variance Function Estimation

Remember that there are two equivalent ways of defining the variance:

$$\mathrm{Var}[X] = \mathbb{E}\left[X^2\right] - (\mathbb{E}[X])^2 = \mathbb{E}\left[(X - \mathbb{E}[X])^2\right] \tag{21.27}$$

The latter is more useful for us when it comes to estimating variance functions. We have already figured out how to estimate means — that's what all this previous work on smoothing and regression is for — and the deviation of a random variable from its mean shows up as a residual.

There are two generic ways to estimate conditional variances, which differ slightly in how they use non-parametric smoothing. We can call these the **squared residuals method** and the **log squared residuals method**. Here is how the first one goes.

1. Estimate $m(x)$ with your favorite regression method, getting $\hat{m}(x)$.
2. Construct the **squared residuals**, $u_i = (y_i - \hat{m}(x_i))^2$.
3. Use your favorite *non-parametric* method to estimate the conditional mean of the u_i, call it $\widehat{q}(x)$.
4. Predict the variance using $\widehat{\sigma}^2_x = \widehat{q}(x)$.

The log-squared residuals method goes very similarly.[4]

1. Estimate $m(x)$ with your favorite regression method, getting $\hat{m}(x)$.
2. Construct the **log squared residuals**, $z_i = \log(y_i - \hat{m}(x_i))^2$.
3. Use your favorite *non-parametric* method to estimate the conditional mean of the z_i, call it $\hat{s}(x)$.
4. Predict the variance using $\widehat{\sigma}^2_x = \exp \widehat{s}(x)$.

The quantity $y_i - \hat{m}(x_i)$ is the i^{th} residual. If $\widehat{m} \approx m$, then the residuals should have mean zero. Consequently the variance of the residuals (which is what we want) should equal the expected squared residual. So squaring the residuals makes sense, and the first method just smoothes these values to get at their expectations.

What about the second method — why the log? Basically, this is a convenience — squares are necessarily non-negative numbers, but lots of regression methods don't easily include constraints like that, and we really don't want to predict negative variances.[5] Taking the log gives us an unbounded range for the regression.

Strictly speaking, we don't need to use non-parametric smoothing for either method. If we had a parametric model for σ^2_x, we could just fit the parametric model to the squared residuals (or their logs). But even if you think you know what the variance function should look like it, why not check it?

[4]I learned it from Wasserman (2006, pp. 87–88).

[5]Occasionally people do things like claiming that gene differences explains more than 100% of the variance in some psychological trait, and so environment and up-bringing contribute negative variance. Some of them — like Alford *et al.* (2005) — even say this with a straight face.

We came to estimating the variance function because of wanting to do weighted least squares, but these methods can be used more generally. It's often important to understand variance in its own right, and this is a general method for estimating it. Our estimate of the variance function depends on first having a good estimate of the regression function

21.5.1 Iterative Refinement of Mean and Variance: An Example

The estimate $\widehat{\sigma}^2_x$ depends on the initial estimate of the regression function $\hat{m}(x)$. But, as we saw when we looked at weighted least squares, taking heteroskedasticity into account can change our estimates of the regression function. This suggests an iterative approach, where we alternate between estimating the regression function and the variance function, using each to improve the other. That is, we take either method above, and then, once we have estimated the variance function $\widehat{\sigma}^2_x$, we re-estimate $\hat{m}$ using weighted least squares, with weights inversely proportional to our estimated variance. Since this will generally change our estimated regression, it will change the residuals as well. Once the residuals have changed, we should re-estimate the variance function. We keep going around this cycle until the change in the regression function becomes so small that we don't care about further modifications. It's hard to give a strict guarantee, but *usually* this sort of iterative improvement will converge.

Let's apply this idea to our example. Figure 21.3b already plotted the residuals from OLS. Figure 21.11 shows those squared residuals again, along with the true variance function and the estimated variance function.

The OLS estimate of the regression line is not especially good ($\widehat{\beta}_0 = 2.59$ versus $\beta_0 = 3$, $\widehat{\beta}_1 = -3.18$ versus $\beta_1 = -2$), so the residuals are systematically off, but it's clear from the figure that spline smoothing of the squared residuals is picking up on the heteroskedasticity, and getting a pretty reasonable picture of the variance function.

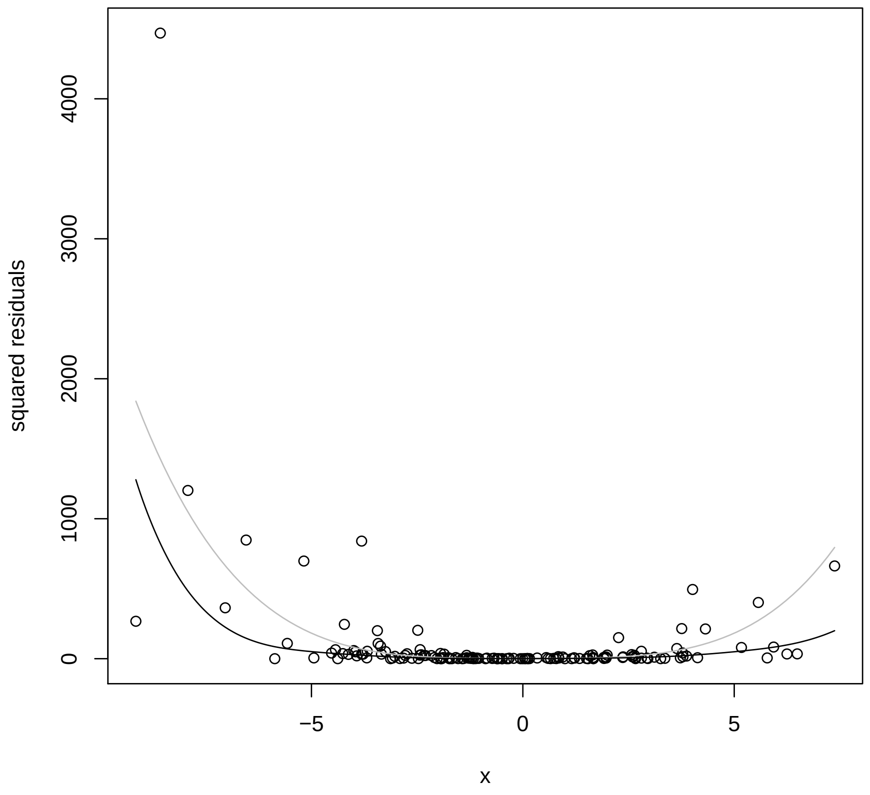
Now we use the estimated variance function to re-estimate the regression line, with weighted least squares.

```
fit.wls1 <- lm(y ~ x, weights = 1/exp(var1$y))
coefficients(fit.wls1)
## (Intercept)           x
##    2.211184   -2.978369
var2 <- smooth.spline(x = x, y = log(residuals(fit.wls1)^2), cv = TRUE)
```

The slope has changed substantially, and in the right direction (Figure 21.12a). The residuals have also changed (Figure 21.12b), and the new variance function is closer to the truth than the old one.

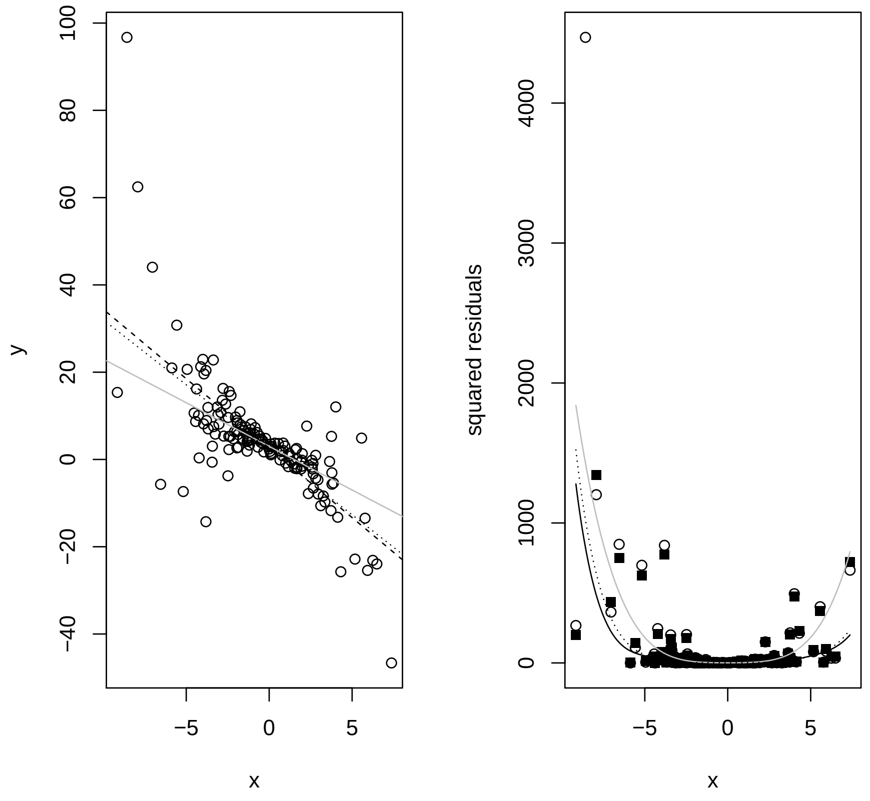
Since we have a new variance function, we can re-weight the data points and re-estimate the regression:

```
fit.wls2 <- lm(y ~ x, weights = 1/exp(var2$y))
coefficients(fit.wls2)
## (Intercept)           x
##    2.243152   -3.041736
var3 <- smooth.spline(x = x, y = log(residuals(fit.wls2)^2), cv = TRUE)
```



```
plot(x, residuals(fit.ols)^2, ylab = "squared residuals")
curve((1 + x^2/2)^2, col = "grey", add = TRUE)
var1 <- smooth.spline(x = x, y = log(residuals(fit.ols)^2), cv = TRUE)
grid.x <- seq(from = min(x), to = max(x), length.out = 300)
lines(grid.x, exp(predict(var1, x = grid.x)$y))
```

FIGURE 21.11: *Points: actual squared residuals from the OLS line. Grey curve: true variance function,* $\sigma_x^2 = (1 + x^2/2)^2$. *Black curve: spline smoothing of the squared residuals.*



```
fit.wls1 <- lm(y ~ x, weights = 1/exp(var1$y))
par(mfrow = c(1, 2))
plot(x, y)
abline(a = 3, b = -2, col = "grey")
abline(fit.ols, lty = "dashed")
abline(fit.wls1, lty = "dotted")
plot(x, (residuals(fit.ols))^2, ylab = "squared residuals")
points(x, residuals(fit.wls1)^2, pch = 15)
lines(grid.x, exp(predict(var1, x = grid.x)$y))
var2 <- smooth.spline(x = x, y = log(residuals(fit.wls1)^2), cv = TRUE)
curve((1 + x^2/2)^2, col = "grey", add = TRUE)
lines(grid.x, exp(predict(var2, x = grid.x)$y), lty = "dotted")
par(mfrow = c(1, 1))
```

FIGURE 21.12: *Left: As in Figure 21.2, but with the addition of the weighted least squares regression line (dotted), using the estimated variance from Figure 21.11 for weights. Right: As in Figure 21.11, but with the addition of the residuals from the WLS regression (black squares), and the new estimated variance function (dotted curve).*

Since we know that the true coefficients are 3 and −2, we know that this is moving in the right direction. If I hadn't told you what they were, you could still observe that the difference in coefficients between `fit.wls1` and `fit.wls2` is smaller than that between `fit.ols` and `fit.wls1`, which is a sign that this is converging.

I will spare you the plot of the new regression and of the new residuals. When we update a few more times:

```
fit.wls3 <- lm(y ~ x, weights = 1/exp(var3$y))
coefficients(fit.wls3)
## (Intercept)           x
##    2.239878   -3.024477
var4 <- smooth.spline(x = x, y = log(residuals(fit.wls3)^2), cv = TRUE)
fit.wls4 <- lm(y ~ x, weights = 1/exp(var4$y))
coefficients(fit.wls4)
## (Intercept)           x
##    2.240906   -3.027914
```

By now, the coefficients of the regression are changing relatively little, and we only have 150 data points, so the imprecision from a limited sample surely swamps the changes we're making, and we might as well stop.

Manually going back and forth between estimating the regression function and estimating the variance function is tedious. We could automate it with a function, which would look something like this:

```
iterative.wls <- function(x, y, tol = 0.01, max.iter = 100) {
    iteration <- 1
    old.coefs <- NA
    regression <- lm(y ~ x)
    coefs <- coefficients(regression)
    while (is.na(old.coefs) || ((max(abs(coefs - old.coefs)) > tol) && (iteration <
        max.iter))) {
        variance <- smooth.spline(x = x, y = log(residuals(regression)^2), cv = TRUE)
        old.coefs <- coefs
        iteration <- iteration + 1
        regression <- lm(y ~ x, weights = 1/exp(variance$y))
        coefs <- coefficients(regression)
    }
    return(list(regression = regression, variance = variance, iterations = iteration))
}
```

This starts by doing an unweighted linear regression, and then alternates between WLS for the getting the regression and spline smoothing for getting the variance. It stops when no parameter of the regression changes by more than `tol`, or when it's gone around the cycle `max.iter` times.[6] This code is a bit too inflexible to be really "industrial strength" (what if we wanted to use a data frame, or a more complex regression formula?), but shows the core idea.

21.6 Correlated Noise and Generalized Least Squares

Sometimes, we might believe the right model is (in matrix form)

$$\mathbf{Y} = \mathbf{X}\beta + \epsilon \tag{21.28}$$
$$\mathbb{E}[\epsilon|\mathbf{X}] = \mathbf{0} \tag{21.29}$$
$$\mathrm{Var}[\epsilon|\mathbf{X}] = \mathbf{\Sigma} \tag{21.30}$$

where the matrix $\mathbf{\Sigma}$ is *not* diagonal. The off-diagonal entries represent covariance in the noise terms, $\mathrm{Cov}\left[\epsilon_i, \epsilon_j\right] = \Sigma_{ij}$. In fact, we should think this is the right model more often than the "usual" linear regression model, which is the special case where $\mathbf{\Sigma} = \sigma^2\mathbf{I}$. There is, after all, no reason in general considerations of probability theory or mathematical modeling to *expect* that fluctuations around a linear model will be uncorrelated. How might we nonetheless estimate β?

One approach is to try to make the noise disappear, by transforming the variables. Suppose we know $\mathbf{\Sigma}$. (We'll come back to where such knowledge might come from

[6]The condition in the `while` loop is a bit complicated, to ensure that the loop is executed at least once. Some languages have an `until` control structure which would simplify this.

later.) Because Σ is a variance matrix, we know it is square, symmetric, and positive-definite. This is enough to guarantee[7] that there is another square matrix, say $\mathbf{s}$, where $\mathbf{s}\mathbf{s}^T = \Sigma$, as it were $\mathbf{s} = \sqrt{\Sigma}$. I bring this fact up because we can use this to make the correlations in the noise go away.

Go back to our model equation, and multiply everything from the left by $\mathbf{s}^{-1}$.

$$\mathbf{s}^{-1}\mathbf{Y} = \mathbf{s}^{-1}\mathbf{X}\beta + \mathbf{s}^{-1}\epsilon$$

This looks like a linear regression of $\mathbf{s}^{-1}\mathbf{Y}$ on $\mathbf{s}^{-1}\mathbf{X}$, with the same coefficients β as our original regression. However, we have improved the properties of the noise. The noise is still zero in expectation,

$$\mathbb{E}\left[\mathbf{s}^{-1}\epsilon|\mathbf{X}\right] = \mathbf{s}^{-1}\mathsf{0} = \mathsf{0}$$

but the covariance has gone away, and all the noise terms have equal variance:

$$\begin{aligned}
\mathrm{Var}\left[\mathbf{s}^{-1}\epsilon|\mathbf{x}\right] &= \mathbf{s}^{-1}\mathrm{Var}\left[\epsilon|\mathbf{x}\right]\mathbf{s}^{-T} && (21.31)\\
&= \mathbf{s}^{-1}\Sigma\mathbf{s}^{-T} && (21.32)\\
&= \mathbf{s}^{-1}\mathbf{s}\mathbf{s}^{T}\mathbf{s}^{-T} && (21.33)\\
&= \mathbf{I} && (21.34)
\end{aligned}$$

(This multiplication by $\mathbf{s}^{-1}$ is the equivalent, for random vectors, of dividing a random variable by its standard deviation, to get something with variance 1.)

To sum up, if we know Σ, we can estimate β by doing an ordinary least squares regression of $\mathbf{s}^{-1}\mathbf{Y}$ on $\mathbf{s}^{-1}\mathbf{X}$. The estimate is

$$\begin{aligned}
\widehat{\beta} &= ((\mathbf{s}^{-1}\mathbf{x})^T\mathbf{s}^{-1}\mathbf{x})^{-1}(\mathbf{s}^{-1}\mathbf{x})^T\mathbf{s}^{-1}\mathbf{y} && (21.35)\\
&= (\mathbf{x}^T\mathbf{s}^{-T}\mathbf{s}^{-1}\mathbf{x})^{-1}\mathbf{x}^T\mathbf{s}^{-T}\mathbf{s}^{-1}\mathbf{y} && (21.36)\\
&= (\mathbf{x}^T\Sigma^{-1}\mathbf{x})^{-1}\mathbf{x}^T\Sigma^{-1}\mathbf{y} && (21.37)
\end{aligned}$$

This looks just like our weighted least squares estimate, only with Σ^{-1} in place of $\mathbf{w}$.

21.6.1 Generalized Least Squares

This resemblance is no mere coincidence. We can write the WLS problem as that of minimizing $(\mathbf{y}-\mathbf{x}\beta)^T\mathbf{w}(\mathbf{y}-\mathbf{x}\beta)$, for a diagonal matrix $\mathbf{w}$. Suppose we try instead to minimize

$$(\mathbf{y}-\mathbf{x}\beta)^T\mathbf{w}(\mathbf{y}-\mathbf{x}\beta)$$

for a *non-diagonal*, but still symmetric and positive-definite, matrix $\mathbf{w}$. This is called a **generalized least squares** (GLS) problem. Every single step we went through before is still valid, because none of it rested on $\mathbf{w}$ being diagonal, so

$$\widehat{\beta}_{GLS} = (\mathbf{x}^T\mathbf{w}\mathbf{x})^{-1}\mathbf{x}^T\mathbf{w}\mathbf{y} \qquad (21.38)$$

[7] Here's one way to do it: invoke the "spectral" or "eigendecomposition" theorem, to write $\Sigma = \mathbf{v}\lambda\mathbf{v}^T$, where $\mathbf{v}$ is the matrix whose columns are the eigenvectors of Σ, and λ is the diagonal matrix of the eigenvalues of Σ. Then if we set $\mathbf{s} = \mathbf{v}\sqrt{\lambda}$, we'd have $\Sigma = \mathbf{s}\mathbf{s}^T$, as desired.

What we have just seen is that if we set $\mathbf{w} = \Sigma^{-1}$, we also get this solution when we transform the variables so as to de-correlate the noise, and then do ordinary least squares. This should at least make it *plausible* that this is a good way to estimate β in the face of correlated noise.

To go beyond plausibility, refer back to §21.3. At no point in our reasoning did we actually rely on $\mathrm{Var}[\epsilon|\mathbf{x}]$ being diagonal. It follows that if we set $\mathbf{w} = \mathrm{Var}[\epsilon|\mathbf{x}]^{-1}$, we get the linear, unbiased estimator of minimum variance. If we believe that the noise is Gaussian, then this is also the maximum likelihood estimator.

21.6.2 Where Do the Covariances Come From?

The soundest way to estimate a covariance would be to repeat the experiment many times, under identical conditions. This corresponds to using repeated measurements to estimate variances. It's simple, it works when we can do it, and there is accordingly little to say about it. Except: there are few situations where we can do it.

When we wanted to estimate the variance function, we could take all the squared residuals for values of x_i around a given x and use that as an estimate of $\sigma^2(x)$. This option is not available to us when we are looking at covariances.

If our measurements are spread out over time or space, it's natural to suppose that there is more covariance between nearby observations than between remote ones. A stronger but more delicate assumption is that of **stationarity**, that the covariance between an observation taken at time 0 and time h is the same as the covariance between time t and time $t + h$, whatever t might be. (And similarly for spatial stationarity.) Call the covariance in at this **lag** or **separation** $\gamma(h)$. We can estimate it by taking pairs of observations where the separation is approximately h, and averaging the products of their residuals.

It is common (though perhaps not wise) to make even stronger assumptions, such as that the covariance decays exponentially with distance, $\gamma(h) = \gamma(0)\rho^h$ or $\gamma(h) = \gamma(0)e^{-h/\tau}$. When we can believe such assumptions, they let us estimate the parameters of the covariance function using the sample covariances across all lags. The estimated covariance function, using all of that data, is much more stable than having many separate sample covariances, one for each lag. Even if the assumptions are, strictly, false, the stability that comes from forcing all the covariances to follow a common model can be desirable, on bias-variance grounds.

21.7 WLS and GLS vs. Specification Errors

When you find that your residuals from an initial model have non-constant variance or are correlated with each other, there are (at least) two possible explanations. One is that the fluctuations around the regression line really are heteroskedastic and/or correlated. In that case, you should try to model that variance and those correlations, and use WLS or GLS. The other explanation is that something is wrong with your model. If there's an important predictor variable which is just missing from your model, for example, then its contribution to the response will be part of your residuals. If that omitted variable is larger in some parts of the data than in others, or if the omitted variable has correlations, then that will make your residuals change in magnitude and be correlated. More subtly, having the wrong functional form for a variable you do include can produce those effects as well.

21.8 Exercises

1. Imagine we are trying to estimate the mean value of Y from a large population. We observe n members of the population, with individual i being included in our sample with a probability proportional to π_i. Show that the sample mean $n^{-1}\sum_{i=1}^{n} y_i$ is *not* a consistent estimator of $\mathbb{E}[Y]$ unless all the π_i are equal. Show that $\left(\sum_{i=1}^{n} y_i/\pi_i\right)/\sum_{i'=1}^{n} 1/\pi_{i'}$ *is* a consistent estimator of $\mathbb{E}[Y]$.
2. Show that the model of Eq. 21.14 has the log-likelihood given by Eq. 21.15
3. Do the calculus to verify Eq. 21.6.
4. Is $w_i = 1$ a necessary as well as a sufficient condition for Eq. 21.3 and Eq. 21.1 to have the same minimum?
5. §21.2.2 showed that WLS gives better parameter estimates than OLS when there is heteroskedasticity, and we know and use the variance. Modify the code for to see which one has better generalization error.

Chapter 22

Variable Selection

22.1 What Variable Selection Is

"Variable selection" means selecting which variables to include in our model (rather than some sort of selection which is itself variable). As such, it is a special case of model selection. People tend to use the phrase "variable selection" when the competing models differ on which variables should be included, but agree on the mathematical form that will be used for each variable — e.g., temperature might or might not be included as a predictor, but there is no question about whether, if it is, we'd use temperature or temperature^2 or $\log$ temperature.

Since variable selection is a special case of model selection, and we've talked extensively about model selection already (see especially Chapter 19), this chapter can be briefer than usual.

Note to self: so why not merge this all into the model-selection chapter?

22.2 Why Variable Selection Using p-Values Is a Bad Idea

When we assume the linear, constant-variance, independent-Gaussian-noise model is completely correct, it is easy to test the hypothesis that any particular coefficient is zero. The (Wald) test statistic is

$$\frac{\hat{\beta}_i}{\widehat{\mathrm{se}}\left[\hat{\beta}_i\right]}$$

and, under the null hypothesis that $\beta_i = 0$, this has a $t_{n-(p+1)}$ distribution, therefore tending to a z (standard-Gaussian) distribution as $n \rightarrow \infty$.

It is very, very tempting, and common, to use the p-values which come from this test to select variables: significant variables get included, insignificant ones do not, ones with smaller p-values (hence larger test statistics) are higher priorities to include than ones with smaller test statistics. This pattern of reasoning shows up over and over again among users of regression, including, I am ashamed to say, not a few statisticians.

The reasons why this is a bad idea were already gone over in Chapter 13, so, again, I will be brief. Let us think about what will tend to make the test statistic larger or smaller, by being more explicit about the denominator:

$$\frac{\hat{\beta}_i}{\frac{\hat{\sigma}}{\sqrt{n\widehat{\mathrm{Var}}[X_i]}}\sqrt{VIF_i}}$$

where $\widehat{\mathrm{Var}}[X_i]$ is the sample variance of the i^{th} predictor variable, and VIF_i is that variables variance-inflation factor (see Chapter 15). What follows from this?

1. Larger coefficients will, all else being equal, have larger test statistics and be more significant ($\hat{\beta}_i$ in the numerator).

2. Reducing the noise around the regression line will increase all the test statistics, and make every variable more significant ($\hat{\sigma}$ in the denominator).

3. Increasing the sample size will increase all the test statistics, and make every variable more significant ($\sqrt{n}$ in the denominator).

4. More variance in a predictor variable will, all else being equal, increase the test statistic and make the variable more significant ($\widehat{\mathrm{Var}}[X_i]$ in the denominator).

5. More correlation between X_i and the other predictors will, all else being equal, decrease the test statistic and make the variable less significant (VIF_i in the denominator).

The test statistic, and thus the p-value, runs together an estimate of the actual size of the coefficient with how well we can measure that particular coefficient. This is exactly the right thing to do if our question is "Can we reliably detect that this coefficient isn't exactly zero?" That is a very, very different question from "Is this variable truly relevant to the response?", or even from "Does including this variable help us predict the response?" Utterly trivial variables can show up as having highly significant coefficients, if the predictor has lots of variance and isn't very correlated with the other predictors. Very important (large-coefficient) variables can be insignificant, when their coefficients can't be measured precisely with our data. *Every* variable whose coefficient isn't exactly zero will eventually (as $n \to \infty$) have an arbitrarily large test statistic, and an arbitrarily small p-value[1]

None of this is even much help in answering the question "Which variables help us predict the response?", let alone "Which variables help us explain the response?"

None of this is fixed by using F-tests on groups of coefficients, rather than t-tests on individual coefficients.

[1]"Oh my God, it's full of stars." — David Bowman, on increasing his sample size to 2001.

22.3 Cross-Validation Instead

If we want to use our models to make predictions, then what we want to know is how well the model will predict new data. The C_p statistic and AIC attempt to estimate this, using how well the model predicted the old data, plus adjustments based on theory. Cross-validation estimates how well the model will predict new data by predicting new data. This is, unsurprisingly, a very good way of estimating how well the model will predict.

The two main forms of cross-validation are leave-one-out, which we have already discussed in detail, and k-fold cross-validation, which we have spent less time on in class but is described in Chapter 19. They each have their strengths and weaknesses (which is why we have both).

- k-fold CV is fast (the model gets fit only k times, and typically k is 5 or 10); it is also "consistent for variable selection", meaning that if one of the models presented to it contains all the relevant predictors, and *only* the relevant predictors, then the probability of picking that right model $\rightarrow 1$ as $n \rightarrow \infty$. On the other hand, it tends to give somewhat worse predictions than leave-one-out, especially when all the models are wrong.

- Leave-one-out can be slow (because the model must be fit n times), *except* for linear regression where there is a short-cut formula. LOOCV is *in*-consistent for variable selection: even with unlimited amounts of data, it tends to include more variables than are necessary, though it will tend to include all the relevant variables. The model it picks tends to have lower prediction errors on new data than those picked by k-fold CV.

As discussed in Chapter 19, C_p and AIC are best seen as approximations to leave-one-out, which avoid the step of re-fitting the model, or even of calculating the short-cut formula (which still involves summing over every data point).

22.4 Stepwise Variable Selection

"Stepwise" or "stagewise" variable selection is a family of methods for adding or removing variables from a model sequentially.

Forward stepwise regression starts with a small model (perhaps just an intercept), considers all one-variable expansions of the model, and adds the variable which is best according to some criterion. This criterion might be "lowest p-value", "highest adjusted R^2", "lowest Mallow's C_p", "lowest AIC", "lowest score under cross-validation", etc. This process is then repeated, always adding one variable at a time, until the criterion stops improving. In **backwards** stepwise regression, we start on the contrary with the largest model we're willing ton contemplate, and keep eliminating variables until we no longer improve. The obvious **forward-backward** or **mixed** stepwise variable selection procedure will contemplating both adding and removing one variable at each step, and take the best step.

In a forward-backward algorithm we could easily add one variable, then add or remove another, and then remove the first variable we'd added. This is because these stepwise algorithms only look at models which are close (one variable away from) the variable we started with[2] In principle, we could just look at all possible linear models based on a given set of variables, and compute our criterion (adjusted R^2, C_p, AIC, LOOCV, etc.) for each one of them; this is called **all-subsets** variable selection, because each model corresponds to a subset of the variables. With p variables there are 2^p possible models, so all-subsets regression becomes, literally, exponentially more time-consuming with more variables; this is the only real justification for the stepwise procedures.

22.4.1 Stepwise Selection in R

The simplest function for stepwise model selection is the `step` function, which is built in to R. It can do forward or backward selection, or both, and you can specify both the smallest model to consider (so those variables are always included), and the largest. It can, however, only use AIC or BIC as the selection criteria.

Here's an example of how it works[3], for the real estate data set from homework 8[4].

```
real.estate <- read.csv("http://www.stat.cmu.edu/~cshalizi/mreg/15/hw/08/real-estate.csv")
# Fit a 'kitchen sink' model But don't try to use the ID numbers as a
# predictor variable!
(realty.lm.all <- lm(Price ~ . - ID, data = real.estate))
##
## Call:
## lm(formula = Price ~ . - ID, data = real.estate)
##
## Coefficients:
##     (Intercept)             Sqft          Bedroom         Bathroom
##      -2.390e+06        1.075e+02       -9.712e+03       -1.067e+02
## Airconditioning           Garage             Pool        YearBuild
##      -1.222e+04        1.732e+04        1.249e+04        1.279e+03
##         Quality              Lot       AdjHighway
##      -5.390e+04        1.422e+00       -2.717e+04
step(realty.lm.all, direction = "backward", trace = 0)
##
## Call:
## lm(formula = Price ~ Sqft + Bedroom + Garage + YearBuild + Quality +
##     Lot, data = real.estate)
##
## Coefficients:
```

[2]As search algorithms, they are "greedy".

[3]The `trace` argument controls how much `step` prints out as it tries various models. Larger values print out more information; the default, `trace=1`, is already a lot. Setting it to zero suppresses this. I urge you to re-run these examples with `trace=1`, but I doing so would substantially lengthen these notes.

[4]But without any attempt at cleaning the data by removing outliers, etc.; this is just to illustrate the syntax, not as a full-scale data analysis.

```
## (Intercept)         Sqft      Bedroom       Garage    YearBuild
##  -2.233e+06    1.093e+02   -1.007e+04    1.665e+04    1.191e+03
##     Quality          Lot
##  -5.223e+04    1.415e+00
```

By comparison to the kitchen sink model, this drops bathrooms, air-conditioning, the pool, and adjacency to highways.

Of course, we could start with a very simple model and expand:

```
(realty.lm.minimal <- lm(Price ~ 1, data = real.estate))
##
## Call:
## lm(formula = Price ~ 1, data = real.estate)
##
## Coefficients:
## (Intercept)
##      277894
step(realty.lm.minimal, scope = list(upper = realty.lm.all, lower = realty.lm.minimal),
    direction = "forward", trace = 0)
##
## Call:
## lm(formula = Price ~ Sqft + Quality + YearBuild + Lot + Garage +
##     Bedroom, data = real.estate)
##
## Coefficients:
## (Intercept)         Sqft      Quality    YearBuild          Lot
##  -2.233e+06    1.093e+02   -5.223e+04    1.191e+03    1.415e+00
##      Garage      Bedroom
##   1.665e+04   -1.007e+04
```

This begins with an intercept-only model, and then adds variables. Here giving a lower limit to the scope is pretty much superfluous, we could just give it an upper limit, but it doesn't hurt.

Of course, we can also ask `step` to consider both adding and subtracting variables:

```
step(realty.lm.minimal, scope = list(upper = realty.lm.all, lower = realty.lm.minimal),
    direction = "both", trace = 0)
##
## Call:
## lm(formula = Price ~ Sqft + Quality + YearBuild + Lot + Garage +
##     Bedroom, data = real.estate)
##
## Coefficients:
## (Intercept)         Sqft      Quality    YearBuild          Lot
##  -2.233e+06    1.093e+02   -5.223e+04    1.191e+03    1.415e+00
##      Garage      Bedroom
##   1.665e+04   -1.007e+04
```

(This just so happens to reach the same answer as only doing forward selection.)
If we want to only consider models which include certain terms, we can do that through changing the lower limit of the scope:

```
(realty.lm.comforts <- lm(Price ~ Pool + Airconditioning, data = real.estate))
##
## Call:
## lm(formula = Price ~ Pool + Airconditioning, data = real.estate)
##
## Coefficients:
##     (Intercept)             Pool  Airconditioning
##          188852            64335           101760
step(realty.lm.comforts, scope = list(upper = realty.lm.all, lower = realty.lm.comforts),
    direction = "both", trace = 0)
##
## Call:
## lm(formula = Price ~ Pool + Airconditioning + Sqft + Quality +
##     YearBuild + Lot + Garage + Bedroom, data = real.estate)
##
## Coefficients:
##     (Intercept)             Pool  Airconditioning             Sqft
##      -2.346e+06        1.279e+04       -1.176e+04        1.080e+02
##         Quality        YearBuild              Lot           Garage
##      -5.388e+04        1.256e+03        1.389e+00        1.724e+04
##         Bedroom
##      -9.756e+03
```

The step function is a simplified version of the function stepAIC in the MASS package, which works very similarly but is more flexible. The leaps package contains an even more flexible function, subsetreg, which tries to determine the lowest-MSE model at any given number of variables, and then lets you chose how to trade the number of parameters against MSE.

22.5 Inference after Selection, Again

The standard inferential statistics (like the p-values on individual coefficients) are only valid if the model is chosen independent of the data being used to calculate them. If there is any sort of data-dependent model selection, whether stepwise variable selection or something else, they are no longer valid. This applies even to eliminating variables because their coefficients are insignificant. If we do go ahead and use the same data twice, once to pick a model and once to test hypotheses about that model, we will get confidence intervals which are systematically too narrow, p-values which are systematically too small, etc. (See Chapter 19 for more discussion, and an example of how doing model selection on pure noise can lead to apparently highly-significant results.)

The easy cure, as discussed in Chapter 19, is to split the data in half at random, and use one part to do model selection and the other half to do inference for your selected model. Again, there is nothing about variable selection which makes this any different.

22.6 Further Reading

In general, all the references for Chapter 19 are relevant again.

For a vivid example of just how badly misleading selecting variables based on statistical significance can be, see Ward *et al.* (2010).

Chapter 23

Trees

This lecture was based entirely on the corresponding chapter of Shalizi (forthcoming). I should either write a new chapter, or omit.

Chapter 24

The Bootstrap I

24.1 Statistical Inference, Assuming Gaussian Noise

Consider our usual linear model

$$\mathbf{Y} = \mathbf{x}\beta + \epsilon$$

For a lot of results, it's enough to assume that

$$\mathbb{E}[\epsilon|\mathbf{x}] = 0$$

and

$$\mathrm{Var}[\epsilon|\mathbf{x}] = \sigma^2\mathbf{I}$$

These assumptions are enough to show the consistency of the least squares estimate

$$\widehat{\beta} = (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{y}$$

The reason is that these assumptions let us write

$$\widehat{\beta} = \beta + (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\epsilon ,$$

i.e., they let us write the estimate as "true value plus weighted sum of the noise terms". Just as with simple linear regression, we can use this trick to get at a lot of properties of $\widehat{\beta}$: we can show that it's unbiased; that it has variance matrix $\sigma^2(\mathbf{x}^T\mathbf{x})^{-1}$; that, from the previous two properties, $\widehat{\beta} \to \beta$ as $n \to \infty$. But these assumptions are not enough to get useful hypothesis tests or confidence intervals.

For those, we have, so far, assumed that

$$\epsilon \sim N(0, \sigma^2\mathbf{I})$$

independent of $\mathbf{x}$. This Gaussian noise assumption is important, because it gives us the distribution of $\widehat{\beta}$:

$$\widehat{\beta} \sim N(\beta, \sigma^2(\mathbf{x}^T\mathbf{x})^{-1})$$

And the reason for *that* is that if ϵ is Gaussian, then $(\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\epsilon$ is a linear transformation of a Gaussian, which is also Gaussian. From knowing that $\widehat{\beta}$ is a Gaussian, everything we've done by way of statistical inference follows: the hypothesis tests, the confidence intervals, the prediction intervals, the F tests for multiple coefficients, etc. Without Gaussian noise, few of the formulas you memorized for the exam are, strictly, correct.

Looking at Q—Q plots of our residuals is only important because we want them to be Gaussian. The only justification for ever contemplating a Box-Cox transformation is that we'd like the noise to be Gaussian. We really have little reason to ever expect Gaussian noise; it's just very useful when it happens.

24.1.1 Other Parametric Distributions of the Noise

Suppose we didn't know think that ϵ was Gaussian, but still believed it was independently and identically distributed (IID). We might, for instance, think it had a "double exponential" (or "Laplacian") distribution, with probability density function

$$f(\epsilon) \propto e^{-|\epsilon|/L} ,$$

or was a scaled t distribution with a certain number ν of degrees of freedom,

$$f(\epsilon) \propto \left(1 + \frac{(x/s)^2}{\nu}\right)^{-(\nu+1)/2}$$

Both of these have heavier tails than the Gaussian distribution, so they can be very useful in practice.

If we think that the deterministic part of the model should still be linear, we still use least squares to get the estimate $\widehat{\beta}$, and it's still true that

$$\widehat{\beta} = \beta + (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\epsilon \tag{24.1}$$

But a linear combination of double-exponential variables is just a mess, as is a linear combination of t-distributions. A family of distributions where adding two random variables gives another distribution in the same family is called **stable**. The Gaussian distributions are stable, but most others aren't, and hence trying to work out an *exact* sampling theory for most other noise distributions, like the one we have for Gaussian noise, is pretty hopeless.

24.1.2 Asymptotic Gaussianity

Still, let's think about the noisy part of Eq. 24.1 some more. It's

$$(\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\epsilon$$

Let's bundle up $(\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T$ as a matrix, call it $\mathbf{k}$. Then

$$\hat{\beta}_i = \beta_i + \sum_{j=1}^{n} k_{ij}\epsilon_j \tag{24.2}$$

If all of the k_{ij} were equal, we'd understand what was going on. After all, the central limit theorem says that when ϵ_j are IID with mean 0 and variance σ^2, their average tends towards a Gaussian distribution:

$$\sum_{j=1}^{n} \frac{1}{n}\epsilon_j \rightsquigarrow N(0, \sigma^2/n)$$

This is true whatever the distribution of the ϵ_j might be, provided, to repeat, that they're IID and they have mean 0 and variance $0 < \sigma^2 < \infty$. If instead of multiplying each ϵ_j by $1/n$ we multiplied them by some other constant, we'd change the variance but still tend towards a Gaussian:

$$\sum_{j=1}^{n} k\epsilon_j \rightsquigarrow N(0, \sigma^2 k^2 n)$$

On this basis, we might *hope* that, as $n \rightarrow \infty$,

$$\sum_{j=1}^{n} k_{ij}\epsilon_j \rightsquigarrow N(0, \sigma^2 \sum_j k_{ij}^2)$$

The difficulty is that the terms in the sum, while still statistically independent, are no longer *identically* distributed. There are central limit theorems which apply to independent, non-identically distributed random variables, with the basic result being that if none of the k_{ij} is too big compared to the others, the sum is indeed asymptotically Gaussian[1].

If the matrix $\mathbf{x}$ doesn't give too much influence to any particular observations, then these central limit theorems usually apply, and we can say that

$$\widehat{\beta} \rightsquigarrow N(\beta, \sigma^2(\mathbf{x}^T\mathbf{x})^{-1}) \quad (24.3)$$

as $n \rightarrow \infty$, if ϵ is non-Gaussian but IID. From there, of course, all the usual formulas would also come to hold as $n \rightarrow \infty$.

How close to infinity does n have to be? You may have been told a bit of folklore which says that the central limit theorem dominates the behavior of a sample mean once $n > 30$. This is badly wrong even for averages, let alone more complicated functions like regression estimates. There is really no upper limit on how big n might have to be before Eq. 24.3 becomes a good approximation.

24.1.3 Summing Up on Gaussian Noise

To sum up, we have two situations:

1. We can assume that the noise is *exactly* Gaussian and independent, and have a nice body of theory for statistical inference, but we hardly ever see that happen.

[1]If you want to follow this up, this is the "Lindeberg" central limit theorem.

2. We can assume that the noise is just independent, and recover the Gaussian theory asymptotically as $n \rightarrow \infty$, but we don't know what to do when n is finite, or even how big n has to get.

Clearly, this is an unsatisfying situation.

24.2 The Sampling Distribution as the Source of All Knowledge about Uncertainty

Our data comes from some distribution, let's say P. We would like to know some property of this distribution, say θ. (We may think of this as a regression coefficient, or the whole coefficient vector, or $\mathbb{E}[Y|X=x]$ for a particular x, etc.) Since we do not know P, we can't just calculate θ. What we can do, however, is draw a sample D from P, and then we calculate some statistic or other, $T(D)$. This serves as our estimate of θ. (The same goes for hypothesis tests, etc.) Because the data D are random, so is T. In fact, the distribution of T, the **sampling distribution** of our statistic, is set by the distribution of D. If we knew the sampling distribution, we'd know basically everything there is to know for statistical inference:

- The bias would be $\mathbb{E}[T]-\theta$
- The standard error would $\sqrt{\text{Var}[T]}$
- Hypothesis tests would come from quantiles of T
- Confidence intervals would come from inverting hypothesis tests

Unfortunately,

- Interesting statistics T are very complicated functions of the data, so their sampling distribution is complicated even if P is simple;
- Realistic distributions P are usually also complicated; and
- We don't know P anyway.

Put these together, and we should be surprised we can ever get nice, useful formulas for the sampling distribution of any statistic, rather than disappointed that we can't make it happen for regression without Gaussian noise.

The reason we can work out the sampling distributions for regression with Gaussian noise is that we (or rather, the ancestors) carefully adjusted the assumptions about the noise, the model, and the statistic *just so*, and everything came together. (E.g., we needed both a linear estimator and a noise distribution which was stable under linear combinations.) What could we do if we can't, or won't, fine-tune all the assumptions?

24.3 The Monte Carlo Principle

What is sometimes called the **Monte Carlo principle** is a general strategy for figuring out the behavior of complicated functions of complicated distributions: it says to simulate it and see what happens[2]. If we have samples $D_1, D_2, \ldots D_b$ from P, and we want to know the expectation of $T(D)$, we can approximate that as

$$\mathbb{E}[T(D)] \approx \frac{1}{b}\sum_{i=1}^{b} T(D_i) \equiv \overline{T}$$

If we want to know the variance, we can approximate that by

$$\mathrm{Var}[T(D)] \approx \frac{1}{b}\sum_{i=1}^{b} (T(D_i) - \overline{T})^2$$

If we want to know the q^{th} quantile of T, we can order the $T(D_i)$ from smallest to largest, and take the qb value as our estimate. If we want an interval which will contain T with probability $1-\alpha$, we order the $T(D_i)$ from smallest to largest, and exclude those with rank from 1 to $b\alpha/2$ on one side, and from $b(1-\alpha/2)$ to b on the other. And so on and so forth. All we need to do to get at any property of any function of the distribution P is to be able to draw from, or simulate, P.

One limitation of this for statistics is that, of course, we don't know the true P.

24.4 The Bootstrap Principle

The **bootstrap principle** is that if we have good approximation $\hat{P}$ to P, we can simulate from $\hat{P}$, and get a good approximation to the sampling distribution we want. That is, we apply the Monte Carlo principle to a distribution ($\hat{P}$) which we hope is close to the distribution we really care about (P).

More specifically, bootstrapping is always an algorithm, which goes, abstractly, as follows:

1. Observe data D, calculate estimate $T(D)$ and get an approximation $\hat{P}$ to P
2. Repeat b times:
   (a) Simulate surrogate data $\tilde{D}$ from $\hat{P}$.
   (b) Calculate $\tilde{T} = T(\tilde{D})$, just as those $\tilde{D}$ were real data
3. Approximate the distribution of T under P with the distribution of $\tilde{T}$ under $\hat{P}$

[2]The name, and to some extent the technique, originated with the physicists designing first the atomic and then the hydrogen bomb. Those designs required calculating the expectations of many elaborate functions of complex distributions (Serber, 1992). Rather than trying to actually do the integrals involved, they just developed efficient ways to sample from the distributions, and computed sample averages (Metropolis *et al.*, 1953). The spirit of "try it, and see what happens" went deep at Los Alamos…

There are a *lot* of variants and refinements, some of which we will cover as this goes on, but in the meanwhile, we really need to be clearer about what "a good approximation $\hat{P}$ to P" might mean.

The two basic options are **model-based** bootstraps and **re-sampling** bootstraps[3]. In a model-based bootstrap, our approximation $\hat{P}$ is a full model of the data generating process, which we've estimated; we then simulate that model. In a re-sampling bootstrap, we treat the sample we observed as our best estimate at the distribution of the whole population, and so we draw a new sample from our original sample — we re-sample it. (If you like, in re-sampling the empirical distribution *is* our model.) Both of these ideas can be made a bit more concrete in the context of regression.

24.5 Bootstraps for Regression

Any regression model can be written as

$$Y = m(X) + \epsilon$$

with the caveat that the noise term ϵ might not have expectation zero[4] or be independent of X. Specifying the true regression function m and the distribution of the noise ϵ, including its dependence on X, gives us the data-generating distribution P.

Depending on what we are willing to believe about the true regression function m and the noise ϵ, we have different ways of coming up with approximations $\hat{P}$ to P, and different ways of simulating from those approximations.

24.5.1 The Linear, Gaussian Bootstrap

The simplest case we could have is where we think all of our usual modeling assumptions hold, so that $m(x) = x\beta$ and $\epsilon \sim N(0, \sigma^2)$, IID and independent of x. Then simulating from the estimated model is very easy.

```
# Simulate from a previously fitted linear model with Gaussian noise Inputs:
# model; data frame Outputs: new data frame with response values replaced
# Presumes: all necessary variables are in data frame
sim.lm.gauss <- function(mdl, df) {
    # What's the response variable called?  Should be the first variable in the
    # vector of all variables
    resp.var <- all.vars(formula(mdl))[1]
    # What value should we expect for the response?
    expect.resp <- predict(mdl, newdata = df)
    # How big is the noise?
    sigma2.mle <- mean(residuals(mdl)^2)
    # Add appropriately-sized Gaussian noise to the response
    response <- expect.resp + rnorm(nrow(df), 0, sqrt(sigma2.mle))
```

[3]Often called these "parametric" and "non-parametric", respectively, but that's not quite as transparent, I think, as the other names.

[4]For instance, if the $m(X)$ is biased.

```
    df[, resp.var] <- response  # Won't change df outside this function!
    return(df)
}
```

What we are doing in this function is creating a (very small!) imaginary or alternative world, the **simulation world** or **bootstrap world**, where we know that the model $Y = x\hat{\beta} + \epsilon$, $\epsilon \sim N(0, \hat{\sigma}^2)$ is exactly true. If $\hat{\beta} \approx \beta$ and $\hat{\sigma}^2 \approx \sigma^2$, and ϵ is Gaussian, then what we see in the simulation world is (close to) representative of what would happen in the real world if we could repeat our experiments many times. The advantage of the simulation world is that it's easy to re-run the simulation many times, whereas repeating the experiment may be expensive, difficult, unethical or flat-out impossible.

Of course, we don't just want to get a new data set, from an new simulation-world experiment; we want to know what we'd have concluded from that experiment. This is also easy.

```
# Re-estimate a linear model on a new data set Inputs: old model; data frame
# Output: new lm object Presumes: data frame contains columns with appropriate
# names
re.lm <- function(mdl, df) {
    return(lm(formula(mdl), data = df))
}
```

Now if we want to get at the sampling distribution of, say, the estimated coefficient vector $\widehat{\beta}$, we just simulate it. We'll need *some* particular initial estimate to work with, so let's try to predict how much a cat's heart will weigh, from the cat's total body weight and its sex:

```
library(MASS)
data(cats)
cats.lm <- lm(Hwt ~ Sex * Bwt, data = cats)
```

We now simulate from our `cats.lm` model many times (10000 is a conveniently small number), re-estimate the coefficients each time, and store the re-estimates in an array:

```
beta.boots <- replicate(10000, coefficients(re.lm(cats.lm, sim.lm.gauss(cats.lm,
    cats))))
```

`beta.boots` is now a 4, 10000 array, with one row for each coefficient, and 10000 columns, because we replicated the simulation 10000 times. Each column is a separate visit to the bootstrap world, where the true value of β is fixed to our initial estimate

$\widehat{\beta}$. Symbolically, it looks like

$$\begin{bmatrix} \tilde{\beta}_{01} & \tilde{\beta}_{02} & \dots & \tilde{\beta}_{0b} \\ \tilde{\beta}_{11} & \tilde{\beta}_{12} & \dots & \tilde{\beta}_{1b} \\ \vdots & \vdots & \vdots & \vdots \\ \tilde{\beta}_{p1} & \tilde{\beta}_{p2} & \dots & \tilde{\beta}_{pb} \end{bmatrix}$$

That being the case, we can get the bias:

```
rowMeans(beta.boots) - coefficients(cats.lm)
## (Intercept)        SexM         Bwt     SexM:Bwt
##  0.02916253 -0.02528605 -0.01176551  0.01050829
```

Each row contains all of our samples for one coefficient estimate, so the mean along each row is our (approximate) expected value of the estimate; we subtract the truth from that to get the bias.

We can also get the standard errors:

```
apply(beta.boots, 1, sd)
## (Intercept)        SexM         Bwt     SexM:Bwt
##   1.8262271   2.0630580   0.7678846   0.8361266
```

There is no `rowSDs` function, but the utility (or meta-) function `apply` lets us take any array (the first argument) and apply any function (the third argument) to either all its rows (middle argument `1`) or all its columns (middle argument `2`), or every entry in the array (middle argument `c(1,2)`). So the incantation above takes the standard deviation of each row.

To get a $1-\alpha$ confidence interval, we (conceptually) take all the values we got for each coefficient, sort them, and discard the lower and upper $\alpha/2$ tails. The appropriate incantation for 95% intervals is

```
apply(beta.boots, 1, quantile, prob = c(0.05/2, 1 - 0.05/2))
##       (Intercept)       SexM      Bwt   SexM:Bwt
## 2.5%   -0.5114128 -8.2469745 1.076132 0.06191073
## 97.5%   6.7126991 -0.1744782 4.107776 3.33522125
```

Now, none of this is actually *necessary* if we assume the truth is linear-and-Gaussian. We know that the bias is zero; we know that the standard deviations come from $\hat{\sigma}^2(\mathbf{x}^T\mathbf{x})^{-1}$; specifically, for the cats, they're

```
coefficients(summary(cats.lm))[, "Std. Error"]
## (Intercept)        SexM         Bwt     SexM:Bwt
##   1.8428394   2.0617552   0.7759022   0.8373255
```

We also know how to calculate the confidence intervals:

```
confint(cats.lm, level = 0.95)
##                  2.5 %      97.5 %
## (Intercept) -0.6620801  6.62470490
## SexM        -8.2416012 -0.08919944
## Bwt          1.1024137  4.17041438
## SexM:Bwt     0.0208271  3.33170228
```

The fact that these numbers are *very* close to what we got by simulation tells us two things:

1. Simulation can work to replace intricate probabilistic mathematics with straightforward (even simple-minded) computations.
2. Simulation is completely redundant in the linear-Gaussian case, where we, and R, know all the formulas.

Simulation comes into its own when the formulas aren't available.

24.5.2 Linear, Non-Gaussian Noise

Suppose that we think the residuals follow some particular non-Gaussian distribution, which we know up to some set of parameters, e.g., a t distribution. (We might have reached this conviction either because of some actual scientific theory, or by staring at the plot of the residuals.) If we know how to estimate the parameters of this noise distribution, and we can simulate from it, then we are in business.

I will illustrate this idea by using a t distribution for the noise in the model of cat's hearts. This is not an especially great model for this data, but it's only meant as an illustration[5]. We'll need to estimate the parameters of the t distribution; this job is already done for us by the function `fitdistr` in the `MASS` library.

```
# Simulate from a previously fitted linear model with t-distributed noise
# Inputs: model; data frame Outputs: new data frame with response values
# replaced Presumes: all necessary variables are in data frame
sim.lm.t <- function(mdl, df) {
    # What's the response variable called?
    resp.var <- all.vars(formula(mdl))[1]
    # What value should we expect for the response?
    expect.resp <- predict(mdl, newdata = df)
    # Estimate the t parameters, using MASS::fitdistr
    stopifnot(require(MASS))  # Make sure the library's available
    # After the example in help(fitdistr)
    mydt <- function(x, s, df) {
        dt(x/s, df)/s
    }
    t.params <- fitdistr(residuals(cats.lm), mydt, start = list(s = 1, df = 50),
        lower = c(0, 1))$estimate
    # Add appropriately-sized t-noise to the response
```

[5] t-distributed noise is a much better idea in areas like finance.

```
    response <- expect.resp + t.params["s"] * rt(nrow(df), df = t.params["df"])
    df[, resp.var] <- response  # Won't change df outside this function
    return(df)
}
```

Having changed the function we use to simulate, absolutely nothing else needs to change:

```
beta.boots2 <- replicate(10000, coefficients(re.lm(cats.lm, sim.lm.t(cats.lm, cats))))
```

```
apply(beta.boots2, 1, quantile, prob = c(0.05/2, 1 - 0.05/2))
##       (Intercept)       SexM      Bwt   SexM:Bwt
## 2.5%   -0.6079851 -8.1955570 1.155966 0.02749805
## 97.5%   6.4952558 -0.1147882 4.162899 3.27903236
```

24.5.3 Resampling Residuals

Suppose we are willing to believe that

$$Y = x\beta + \epsilon$$

and even that ϵ is independent of x, but not that we have any good idea about what the distribution of ϵ is. What are we to do?

Well, it will still be true that our residuals will give us an estimate of what the noise ϵ looks like — of what the true noise distribution is. We can use that. When we generate new data, we'll take $x\widehat{\beta} + \tilde{\epsilon}$, where $\tilde{\epsilon}$ is our simulated noise. Every time we need a value for $\tilde{\epsilon}$, we'll go to our vector of residuals and draw a random value from it — we will re-sample the residuals, with replacement. Here's how it works computationally:

```
# Simulate from a previously fitted linear model, resampling residuals Inputs:
# model; data frame Outputs: new data frame with response values replaced
# Presumes: all necessary variables are in data frame
sim.lm.residuals <- function(mdl, df) {
    # What's the response variable called?
    resp.var <- all.vars(formula(mdl))[1]
    # What value should we expect for the response?
    expect.resp <- predict(mdl, newdata = df)
    # Resample the residuals
    new.noise <- sample(residuals(mdl), size = length(expect.resp), replace = TRUE)
    # Add new noise to the expected response
    response <- expect.resp + new.noise
    df[, resp.var] <- response  # Won't change df outside this function
    return(df)
}
```

When (as is usually the case) we want the re-sample to be exactly as large as the original sample, we can define a little convenience function:

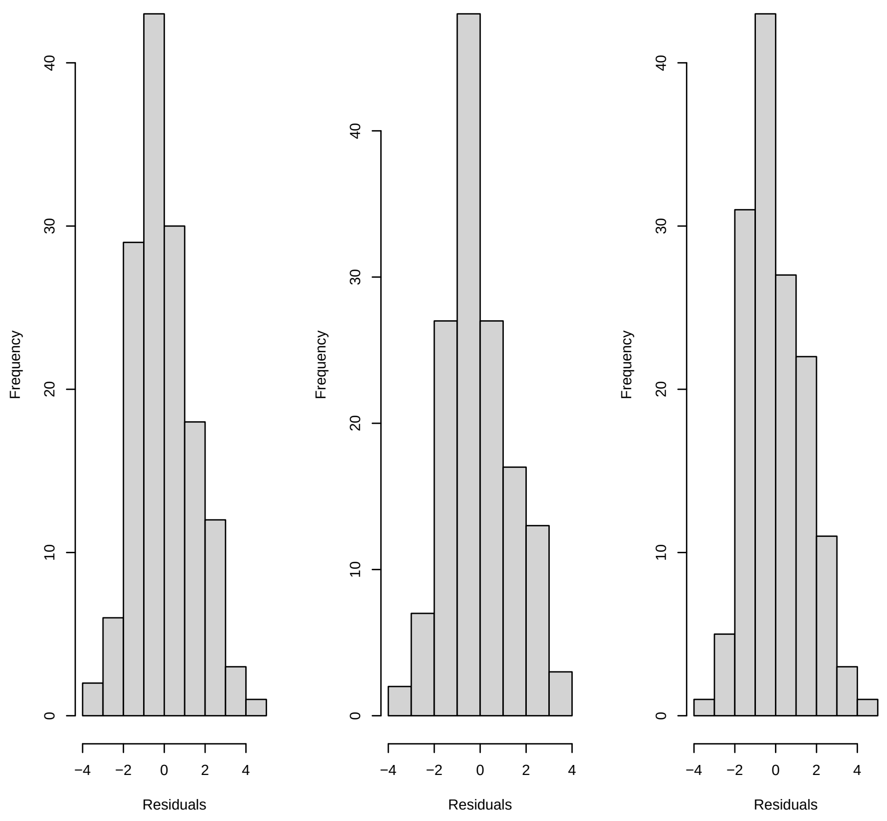
```
resample <- function(x) {
    sample(x, size = length(x), replace = TRUE)
}
```

(Note that, as written, this only works for vectors.) Let's see what kind of things this does to a short vector:

```
head(residuals(cats.lm))
##          1          2          3          4          5          6
## -1.2541405 -0.8541405  1.2458595 -1.3177819 -1.2177819 -0.9177819
resample(head(residuals(cats.lm)))
##          2          5          3          5          4          5
## -0.8541405 -1.2177819  1.2458595 -1.2177819 -1.3177819 -1.2177819
resample(head(residuals(cats.lm)))
##          3          1          2          1          3          3
##  1.2458595 -1.2541405 -0.8541405 -1.2541405  1.2458595  1.2458595
```

Every time we run this function, we get different results; some samples get picked more than once, some don't get picked at all. When we look at the over-all distribution of each re-sample, it's somewhat less diverse than the sample we took it from, just as the sample is less diverse than the population, but it has the same over-all shape.

```
par(mfrow = c(1, 3))
hist(residuals(cats.lm), main = "", xlab = "Residuals")
hist(resample(residuals(cats.lm)), main = "", xlab = "Residuals")
hist(resample(residuals(cats.lm)), main = "", xlab = "Residuals")
```



With this understood, once we have our simulator for resampling residuals, we have to make absolutely no changes to how we use it:

```
beta.boots3 <- replicate(10000, coefficients(re.lm(cats.lm, sim.lm.residuals(cats.lm,
    cats))))
```

```
apply(beta.boots3, 1, quantile, prob = c(0.05/2, 1 - 0.05/2))
##        (Intercept)       SexM      Bwt   SexM:Bwt
## 2.5%    -0.6181038 -8.1326972 1.162809 0.02503376
## 97.5%    6.5230192 -0.1336846 4.160795 3.28090168
```

24.5.4 Resampling Cases

The last bootstrap we'll look at, here, is what's variously called the **case** (or **cases**) bootstrap, or the **pairs** bootstrap, or, more rarely, the **rows** bootstrap. The idea here is that our data consists of cases, each of which contains the predictor variables and the response variables, and we'll re-sample those whole points. The other names arise from thinking of each data point as a pair (x, y), or as a row in a data frame. We're using as our approximation $\hat{P}$ to the data-generating distribution P the joint empirical distribution of the predictors and the response.

The simulator is now sheer elegance in its simplicity:

```
# Re-sample the rows of a data frame Inputs: the data frame Output: a new data
# frame, contain a random sample, with replacement, of rows from the input
resample.data.frame <- function(df) {
    df[resample(1:nrow(df)), ]
}
```

Notice that we don't use the estimated model here at all in the simulation — the procedure is quite agnostic as to whether our model is good or bad.

Having the simulator in hand, we can use it just like the others, and do inference using the simulation just like the others:

```
beta.boots4 <- replicate(10000, coefficients(re.lm(cats.lm, resample.data.frame(cats))))
```

```
apply(beta.boots4, 1, quantile, prob = c(0.05/2, 1 - 0.05/2))
##       (Intercept)       SexM      Bwt  SexM:Bwt
## 2.5%    0.1610193 -7.8819191 1.450448 0.2299905
## 97.5%   5.8309568 -0.5529635 3.839225 3.1292420
```

Resampling cases makes only very weak assumptions about the data-generating distribution, that all data points ((x, y) pairs) are independent and identically distributed. It does not assume that any linear regression is correct[6], or that the noise is independent of x, or has constant variance.

24.6 Error in the Bootstrap, and Which Bootstrap When?

The bootstrap, remember, is a way of calculating the sampling distribution of T, under the true data-generating distribution P. There are two main sources of error in this calculation:

Simulation We'd like to see the full distribution of our statistic T under $\hat{P}$, but instead we only run b simulations under $\hat{P}$.

Approximation We're simulating from $\hat{P}$ rather than P.

Simulation error (or "Monte Carlo error") is easy to grasp, and in principle easy to control: the more simulations we run, the better. As $b \to \infty$, the simulation goes

[6]If the linear model *is* wrong, then we're doing statistical inference on the coefficients in the best linear approximation to the true regression function $m(x)$.

to zero. The only reason to ever restrict b is that each simulation does have some cost, in time if nothing else. In fact, for (most) inferential statistics, we can expect the simulation error to be $O(1/\sqrt{b})$, so if we keep increasing b we will experience *diminishing* returns, but never negative returns.

Approximation error comes from the fact that $\hat{P}$ is not P. This can itself be broken into two parts: estimation error (roughly, variance), and systematic distortion (roughly, bias). Estimation error arises because we only have a finite amount of data, say n observations, with which to estimate $\hat{P}$. Even if we resample cases, and so use the empirical distribution as our $\hat{P}$, we still only have n samples from the full population, which isn't all of it. Generally, estimation error will shrink to zero as $n \to \infty$, but it may shrink at different rates for different approximations. Systematic distortion is basically the approximation error which would be left even if we had infinite data — it comes from using a linear-Gaussian simulation when reality isn't linear or Gaussian, or resampling residuals when the noise is really heteroskedastic.

There is a trade-off when it comes to the two kinds of approximation error. The more we constrain $\hat{P}$ in advance of seeing any data, the stronger the assumption we put on it, the less we have to estimate, and so the smaller the estimation error. But, the true P doesn't obey those constraints, if our assumptions are wrong, the bigger the systematic distortion we're introducing. If our assumptions are right, using a more constrained $\hat{P}$ is pure advantage — basically, we're not wasting data figuring out that the constraints hold — but if those assumptions are wrong, they can easily make things worse.

Which bootstrap to use, then, depends on how strongly you trust your modeling assumptions.

- If you believe that the regression is linear and you know the distribution of the noise, use the fully model-based bootstraps.
- If you believe that the regression is linear and the noise is independent of x, use resampling of residuals.
- If you are unwilling to believe that the noise is independent of x, and/or that the regression is truly linear, use resampling of cases.

(We'll cover the situation where you don't think the truth is linear but you are, somehow, convinced the noise is Gaussian when we go over fitting nonlinear models in 402.)

How do we tell? Well, in the first situation, all of the diagnostics we've been doing should look good, including the appropriate Q-Q plot. In the second situation, while the residuals should have the same distribution for all x, we don't care what that distribution is. Therefore when we plot residuals against predictors and fitted values, everything should look random, but not necessarily Gaussian, and the Q-Q plot need show no particular shape. In the third situation, by resampling cases we're still assuming independence across data points, so we should try to check that, but that's about all that we do need to check.

24.7 Further Reading

The bootstrap was introduced, by that name, by Efron (1979), in a remarkably accessible paper which is still worth reading. Related ideas, such as the "jackknife" (omit one data point, re-estimate, and look at the variance over all such re-estimates) go back at least to the 1940s, though they weren't systematically developed, or applied, until computing power got cheap enough to make something like the bootstrap feasible. A good systematic textbook is Davison and Hinkley (1997).

For the validity of case resampling even when all the usual linear-Gaussian assumptions fail, see, e.g., Buja *et al.* (2014). (That paper also shows how this bootstrap does the same job as the "robust standard errors" of econometrics.)

Next time, we will look at some of the techniques for reducing approximation error in bootstrap calculations, bootstrap prediction intervals, and what sorts of things the bootstrap can't do; all of these are also covered in Davison and Hinkley (1997).

Chapter 25

The Bootstrap, II

25.1 Improving the Bootstrap: Pivoting

Note to self: currently a bit of a stub; merge this into previous chapter?

In Chapter 24, I said there were three causes of error when we use the bootstrap to do statistical inference, e.g., to calculate a confidence interval. These were (1) using only a finite number of simulations; (2) simulating from an approximate distribution $\hat{P}$ which systematically distorts the true, data-generating distribution P; and (3) having only a finite amount of data with which to estimate $\hat{P}$.

Increasing the number of simulations we run is simple (in principle), but does nothing about using a $\hat{P}$ which is not the same as the true P. We can't do much about the systematic error issue either, unless we find better models. But there are tricks for addressing the third issue — of trying to bring $\hat{P}$ closer to P at a given amount of data. One of the most important of these is called **pivoting**, and it involves changing, slightly, what we look at in the simulations.

We have a certain estimate of our quantity of interest from the data, $\hat{\theta}$; this is a random variable with a distribution governed by P. When we run the bootstrap, we get estimates on the bootstrap data, $\tilde{\theta}$, with a distribution governed by $\hat{P}$. So far, we've been saying that the distribution of $\tilde{\theta}$ should be approximately the same as the distribution of $\hat{\theta}$. In symbols,

$$\mathscr{D}(\hat{\theta}) \approx \mathscr{D}(\tilde{\theta})$$

As the number of actual observations grows, these two distributions should indeed converge. The key observation to pivoting is that, at the same n, the distribution of estimation *errors* is often closer than the distribution of *estimates*:

$$\mathscr{D}(\hat{\theta} - \theta_0) \approx \mathscr{D}(\tilde{\theta} - \hat{\theta})$$

In words: the distribution of bootstrap estimates around our real-data estimate is a good approximation to the distribution of estimates around the truth. In fact, it's usually better than using the distribution of bootstrap values to approximate the distribution of estimates.

Confidence Intervals This doesn't change how we get a bootstrap standard error, or a bias. But it does change our bootstrap confidence intervals. Let's say the quantiles of $\tilde{\theta}$ are $q_{\alpha/2}$ and $q_{1-\alpha/2}$. In our first crude bootstrap[1], we'd use the range $[q_{\alpha/2}, q_{1-\alpha/2}]$ as our $1-\alpha$ confidence interval for θ. Here's how we come up with a better interval using pivoting:

$$\begin{aligned}
1-\alpha &= \hat{P}(q_{\alpha/2} \leq \tilde{\theta} \leq q_{1-\alpha/2}) && (25.1)\\
&= P(q_{\alpha/2} - \hat{\theta} \leq \tilde{\theta} - \hat{\theta} \leq q_{1-\alpha/2} - \hat{\theta}) && (25.2)\\
&\approx P(q_{\alpha/2} - \hat{\theta} \leq \hat{\theta} - \theta_0 \leq q_{1-\alpha/2} - \hat{\theta}) && (25.3)\\
&= P(q_{\alpha/2} - 2\hat{\theta} \leq -\theta_0 \leq q_{1-\alpha/2} - 2\hat{\theta}) && (25.4)\\
&= P(2\hat{\theta} - q_{1-\alpha/2} \leq \theta_0 \leq 2\hat{\theta} - q_{\alpha/2}) && (25.5)\\
&= P(\hat{\theta} + (\hat{\theta}) - q_{1-\alpha/2}) \leq \theta_0 \leq \hat{\theta} + (\hat{\theta} - q_{\alpha/2})) && (25.6)
\end{aligned}$$

The third line is where we use the bootstrap principle, and pivoting: the distribution of $\tilde{\theta}$ around $\hat{\theta}$ should be close to the distribution of $\hat{\theta}$ around θ_0. The rest is just book-keeping, where in the last line I've re-written it so it looks more like the confidence intervals we're used to seeing for means or for regression coefficients.

25.2 Improving the Bootstrap: Studentizing and the Double Bootstrap

25.3 A Hint of Theory

25.3.1 Why the Bootstrap Works

25.3.2 When the Bootstrap Fails

[1]Often called the "quantile" or "percentile" bootstrap.

Appendix A

Problem Sets

Weekly problem sets were an essential part of the class. Add them!

Appendix B

TODO

- Fix URLs for data sets to something not so class-specific
- Improve cross-chapter cross-references (to individual sections or even pages, not just the chapter as a whole)
- Add lasso material to chapter on variable selection
- Number all equations
- Fix collisions between captions and footers

B.1 Outline to Be Revised To

This is just an aid to my own memory — CRS

1. Linear Least Squares with One Predictor
   (a) Optimal Linear Prediction
   (b) Estimation by Least Squares
   (c) Diagnostics
   (d) Outliers and Influential Points
   (e) Parametric Inference: Bootstrap and Asymptotics
   (f) Predictive Inference
   (g) Model Selection
   (h) Smoothing Splines
   (i) Gaussian Noise Theory
2. Linear Least Squares with More Than One Predictor
   (a) Linear Models in Matrix Form
   (b) Estimation by Least Squares
   (c) Polynomial Terms, Categorical Predictors, Interactions

   (d) Diagnostics
   (e) Collinearity
   (f) Influential Points and Outliers
   (g) Parametric Inference: Bootstrap and Asymptotics
   (h) Predictive Inference
   (i) Variable Selection
   (j) Linear Prediction and Estimation for Spatio-Temporal Data (Forecasting, Kriging, Filtering)
   (k) Additive Models
   (l) Gaussian Noise Theory

3. Beyond Ordinary Least Squares
   (a) Weighted Least Squares and Heteroskedasticity
   (b) Generalized Least Squares and Correlated Noise
   (c) Ridge Regression
   (d) The Lasso
   (e) Generalized Linear Models
   (f) Regression Trees

4. Let the Dead Bury the Dead
   (a) What F and Wald Tests Actually Test
   (b) R^2: Distraction or Nuisance?
   (c) Transforming for Gaussianity
   (d) ANOVA Tables

Acknowledgments

Thanks go first of all to my students in 36-401, who put up with learning from first drafts posted twice a week. I am also grateful to my colleague Prof. Rebecca Nugent, who taught 401 for many years before me, for access to her teaching materials, and permission to borrow some of them for use in my class (none of her materials appear in this text, however), as well as for her sense of humor about the students' occasional "shenanigans", and her forbearance towards my own foibles. More broadly, I am indebted to my colleagues in the Statistics Department at CMU, for creating a program where a class like this could be taught to undergraduates.

For specific corrections, beyond the students in 401, I thank Martin Buttenschön, David Darmon, Dr. Niels Hagenbuch, Justin Lazarow, Dr. Michael Lösler, and Pedro Enrique Passos. All remaining errors and lapses are, of course, my fault.

During the main period when I wrote these notes, I was partially supported by grants from the Institute for New Economic Thinking (INO1400020) and from the National Science Foundation (DMS1418124). Those were intended to support my research, but money is fungible, so they have my thanks — but of course neither they, nor my regular employer, is in any way responsible for my opinions here.

Bibliography

Adler, Daniel, Duncan Murdoch and others (2014). `rgl`*: 3D visualization device system (OpenGL)*. URL `http://CRAN.R-project.org/package=rgl`. R package version 0.95.1201.

Akaike, Hirotugu (1973). "Information Theory and an Extension of the Maximum Likelihood Principle." In *Proceedings of the Scond International Symposium on Information Theory* (B. N. Petrov and F. Caski, eds.), pp. 267–281. Budapest: Akademiai Kiado. Reprinted in (Akaike, 1998, pp. 199–213).

— (1998). *Selected Papers of Hirotugu Akaike*. Berlin: Springer-Verlag. Edited by Emanuel Parzen, Kunio Tanabe and Genshiro Kitagawa.

Alford, J. R., C. L. Funk and J. R. Hibbibng (2005). "Are Political Orientations Genetically Transmitted?" *American Political Science Review*, **99**: 153–167.

Anderson, Chris (June 2008). "The End of Theory: The Data Deluge Makes the Scientific Method Obsolete." *Wired*, **16(17)**. URL `http://www.wired.com/2008/06/pb-theory/`.

Anderson, Norman H. and James Shanteau (1977). "Weak inference with linear models." *Psychological Bulletin*, **84**: 1155–1170. doi:10.1037/0033-2909.84.6.1155.

Ashby, F. Gregory (2011). *Statistical Analysis of fMRI Data*. Cambridge, Massachusetts: MIT Press.

Axler, Sheldon (1996). *Linear Algebra Done Right*. Undergraduate Texts in Mathematics. Berlin: Springer-Verlag.

Badii, Remo and Antonio Politi (1997). *Complexity: Hierarchical Structures and Scaling in Physics*. Cambridge, England: Cambridge University Press.

Benjamini, Yoav and Yosef Hochberg (1995). "Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing." *Journal of the Royal Statistical Society B*, **57**: 289–300. URL `http://www.math.tau.ac.il/~ybenja/MyPapers/benjamini_hochberg1995.pdf`.

Bennett, Craig M., Abigail A. Baird, Michael B. Miller and George L. Wolford (2010). "Neural correlates of interspecies perspective taking in the post-mortem Atlantic

Salmon: An argument for multiple comparisons correction." *Journal of Serendipitous and Unexpected Results*, **1**: 1–5. URL `http://prefrontal.org/files/posters/Bennett-Salmon-2009.pdf`.

Berk, Richard A. (2004). *Regression Analysis: A Constructive Critique*. Thousand Oaks, California: Sage.

Bettencourt, Luís M. A., José Lobo, Dirk Helbing, Christian Künhert and Geoffrey B. West (2007). "Growth, Innovation, Scaling, and the Pace of Life in Cities." *Proceedings of the National Academy of Sciences (USA)*, **104**: 7301–7306. doi:10.1073/pnas.0610172104.

Birnbaum, Michael H. (1973). "The Devil Rides Again: Correlation as an Index of Fit." *Psychological Bulletin*, **79**: 239–242. doi:10.1037/h0033853.

Bühlmann, Peter (2014). "High-Dimensional Statistics with a View Toward Applications in Biology." *Annual Review of Statistics and Its Applications*, **1**: 255–278. doi:10.1146/annurev-statistics-022513-115545.

Bühlmann, Peter and Sara van de Geer (2011). *Statistics for High-Dimensional Data: Methods, Theory and Applications*. Berlin: Springer-Verlag.

Buja, Andreas, Richard Berk, Lawrence Brown, Edward George, Emil Pitkin, Mikhail Traskin, Linda Zhao and Kai Zhang (2014). "Models as Approximations: A Conspiracy of Random Regressors and Model Deviations Against Classical Inference in Regression." arxiv:1404.1578. URL `http://arxiv.org/abs/1404.1578`.

Casella, George and R. L. Berger (2002). *Statistical Inference*. Belmont, California: Duxbury Press, 2nd edn.

Cramér, Harald (1945). *Mathematical Methods of Statistics*. Uppsala: Almqvist and Wiksells.

Cule, Erika (2014). `ridge`*: Ridge Regression with automatic selection of the penalty parameter*. URL `http://CRAN.R-project.org/package=ridge`. R package version 2.1-3.

Davison, A. C. and D. V. Hinkley (1997). *Bootstrap Methods and their Applications*. Cambridge, England: Cambridge University Press.

Dhillon, Paramveer S., Dean P. Foster, Sham M. Kakade and Lyle H. Ungar (2013). "A Risk Comparison of Ordinary Least Squares vs Ridge Regression." *Journal of Machine Lerning Research*, **14**: 1505–1511. URL `http://jmlr.org/papers/v14/dhillon13a.html`.

DuMouchel, William H. and Greg J. Duncan (1983). "Using Sample Survey Weights in Multiple Regression Analyses of Stratified Samples." *Journal of the American Statistical Association*, **78**: 535–543. URL `http://www.jstor.org/stable/2288115`.

Efron, Bradley (1979). "Bootstrap Methods: Another Look at the Jackknife." *Annals of Statistics*, **7**: 1–26. URL `http://projecteuclid.org/euclid.aos/1176344552`.

Feller, William (1957). *An Introduction to Probability Theory and Its Applications*, vol. I. New York: Wiley, 2nd edn.

Feynman, Richard P. (1985). *QED: The Strange Theory of Light and Matter*. Princeton, New Jersey: Princeton University Press.

Freedman, David A. (1983). "A Note on Screening Regression Equations." *The American Statistician*, **37**: 152–155. doi:10.1080/00031305.1983.10482792.

Galambos, Janos and Italo Simonelli (1996). *Bonferroni-type Inequalities with Applications*. Berlin: Springer-Verlag.

Gelman, Andrew (2005). "Analysis of Variance — Why It Is More Important than Ever." *Annals of Statistics*, **33**: 1–53. URL `http://projecteuclid.org/euclid.aos/1112967698`. doi:10.1214/009053604000001048.

Genovese, Christopher and Larry Wasserman (2004). "A Stochastic Process Approach to False Discovery Control." *Annals of Statistics*, **32**: 1035–1061. URL `http://projecteuclid.org/euclid.aos/1085408494`. doi:10.1214/009053604000000283.

Gouriéroux, Christian and Alain Monfort (1989/1995). *Statistics and Econometric Models*. Themes in Modern Econometrics. Cambridge, England: Cambridge University Press. Translated by Quang Vuong from *Statistique et modèles économétriques*, Paris: Économica.

Hamilton, Richard F. (1996). *The Social Misconstruction of Reality: Validity and Verification in the Scholarly Community*. New Haven: Yale University Press.

Hoerl, Arthur E. and Robert W. Kennard (1970). "Ridge Regression: Biased Estimation for Nonorthogonal Problems." *Technometrics*, **12**. URL `http://www.jstor.org/pss/1267351`.

Hoover, Kevin D. and Mark V. Siegler (2008). "Sound and Fury: McCloskey and Significance Testing in Economics." *Journal of Economic Methodology*, **15**: 1–37. URL `http://hdl.handle.net/10161/2045`. doi:10.1080/13501780801913298.

Klein, Dan (2001). "Lagrange Multipliers without Permanent Scarring." Online tutorial. URL `http://dbpubs.stanford.edu:8091/~klein/lagrange-multipliers.pdf`.

Li, Ming and Paul M. B. Vitányi (1997). *An Introduction to Kolmogorov Complexity and Its Applications*. New York: Springer-Verlag, 2nd edn.

Li, Qi and Jeffrey Scott Racine (2007). *Nonparametric Econometrics: Theory and Practice*. Princeton, New Jersey: Princeton University Press.

Low-Décarie, Etienne, Corey Chivers and Monica Granados (2014). "Rising complexity and falling explanatory power in ecology." *Frontiers in Ecology and the Environment*, **12**: 412–418. doi:10.1890/130230.

Mayo, Deborah G. (1996). *Error and the Growth of Experimental Knowledge*. Chicago: University of Chicago Press.

Mayo, Deborah G. and D. R. Cox (2006). "Frequentist Statistics as a Theory of Inductive Inference." In *Optimality: The Second Erich L. Lehmann Symposium* (Javier Rojo, ed.), pp. 77–97. Bethesda, Maryland: Institute of Mathematical Statistics. URL `http://arxiv.org/abs/math.ST/0610846`.

Mayo, Deborah G. and Aris Spanos (2006). "Severe Testing as a Basic Concept in a Neyman-Pearson Philosophy of Induction." *The British Journal for the Philosophy of Science*, **57**: 323–357. doi:10.1093/bjps/axl003.

McCloskey, D. N. (2002). *The Secret Sins of Economics*. Chicago: Prickly Paradigm Press. URL `www.prickly-paradigm.com/paradigm4.pdf`.

McCloskey, D. N. and S. T. Ziliak (1996). "The Standard Error of Regressions." *Journal of Economic Literature*, **34**: 97–114.

Metropolis, N., A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller and E. Teller (1953). "Equations of State Calculations by Fast Computing Machines." *Journal of Chemical Physics*, **21**: 1087–1092. doi:10.1063/1.1699114.

Murdoch, Duncan and E. D. Chow (2013). `ellipse`*: Functions for drawing ellipses and ellipse-like confidence regions*. URL `http://CRAN.R-project.org/package=ellipse`. R package version 0.3-8.

Neyman, Jerzy (1937). "Outline of a Theory of Statistical Estimation Based on the Classical Theory of Probability." *Philosophical Transactions of the Royal Society of London A*, **236**: 333–380. doi:10.1098/rsta.1937.0005.

Petersen, Kaare Brandt and Michael Syskind Pedersen (2012). *The Matrix Cookbook*. Tech. rep., Technical University of Denmark, Intelligent Signal Processing Group. URL `http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=3274`.

Pitman, E. J. G. (1979). *Some Basic Theory for Statistical Inference*. London: Chapman and Hall.

Quiñonero-Candela, Joaquin, Masashi Sugiyama, Anton Schwaighofer and Neil D. Lawrence (eds.) (2009). *Dataset Shift in Machine Learning*. Cambridge, Massachusetts: MIT Press.

Ruelle, David (1991). *Chance and Chaos*. Princeton, New Jersey: Princeton University Press.

Salmon, Wesley C. (1984). *Scientific Explanation and the Causal Structure of the World*. Princeton: Princeton University Press.

Schervish, Mark J. (1995). *Theory of Statistics*. Berlin: Springer-Verlag.

Schwarz, Gideon (1978). “Estimating the Dimension of a Model.” *Annals of Statistics*, **6**: 461–464. URL `http://projecteuclid.org/euclid.aos/1176344136`.

Seber, George A. F. and Alan J. Lee (2003). *Linear Regression Analysis*. New York: Wiley, 2nd edn.

Serber, Robert (1992). *The Los Alamos Primer: The First Lectures on How to Build the Atomic Bomb*. Berkeley: University of California Press. Annotated by Robert Serber; edited and with an introduction by Richard Rhodes.

Shalizi, Cosma Rohilla (forthcoming). *Advanced Data Analysis from an Elementary Point of View*. Cambridge, England: Cambridge University Press. URL `http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV`.

Shmueli, Galit (2010). “To Explain or to Predict?” *Statistical Science*, **25**: 289–310. doi:10.1214/10-STS330.

Simonoff, Jeffrey S. (1996). *Smoothing Methods in Statistics*. Berlin: Springer-Verlag.

Smith, Leonard (2007). *Chaos: A Very Short Introduction*. Oxford: Oxford University Press.

Swift, Jonathan (1726). *Gulliver’s Travels*. London: Benjamin Motte. URL `http://www.gutenberg.org/ebooks/829`. Originally published as *Travels into Several Remote Nations of the World. In Four Parts. By Lemuel Gulliver, First a Surgeon, and then a Captain of several Ships*.

Thompson, Wilbur R. (1968). *Preface to Urban Economics*. Baltimore: Johns Hopkins University Press.

Tukey, John W. (1954). “Unsolved Problems of Experimental Statistics.” *Journal of the American Statistical Association*, **49**: 706–731. URL `http://www.jstor.org/pss/2281535`.

Tutz, Gerhard (2012). *Regression for Categorical Data*. Cambridge, England: Cambridge University Press.

van der Vaart, A. W. (1998). *Asymptotic Statistics*. Cambridge, England: Cambridge University Press.

Vuong, Quang H. (1989). “Likelihood Ratio Tests for Model Selection and Non-Nested Hypotheses.” *Econometrica*, **57**: 307–333. URL `http://www.jstor.org/pss/1912557`.

Wainwright, Martin J. (2014). “Structured Regularizers for High-Dimensional Problems: Statistical and Computational Issues.” *Annal Review of Statistics and Its Applications*, **1**: 233–253. doi:10.1146/annurev-statistics-022513-115643.

Ward, Michael D., Brian D. Greenhill and Kristin M. Bakke (2010). "The Perils of Policy by p-value: Predicting Civil Conficts." *Journal of Peace Research*, **47**: 363–375. doi:10.1177/0022343309356491.

Wasserman, Larry (2006). *All of Nonparametric Statistics*. Berlin: Springer-Verlag.

Weisburd, David and Alex R. Piquero (2008). "How Well Do Criminologists Explain Crime? Statistical Modeling in Published Studies." *Crime and Justice*, **37**: 453–502. doi:10.1086/524284.

Wilks, S. S. (1938). "The Large Sample Distribution of the Likelihood Ratio for Testing Composite Hypotheses." *Annals of Mathematical Statistics*, **9**: 60–62. URL `http://projecteuclid.org/euclid.aoms/1177732360`. doi:10.1214/aoms/1177732360.

Winship, Christopher and Robert D. Mare (1984). "Regression Models with Ordinal Variables." *American Sociological Review*, **49**: 512–525. URL `http://scholar.harvard.edu/files/cwinship/files/asr_1984.pdf`.