Lecture 5: The Method of Least Squares for Simple Linear Regression
36-401, Fall 2017, Section B

Contents

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.

[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.

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

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

In particular, we obtained the following results:

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

$$\overline{y} - \hat{\beta}_0 - \hat{\beta}_1 \overline{x} = 0 \tag{2}$$

$$\overline{xy} - \hat{\beta}_0 \overline{x} - \hat{\beta}_1 \overline{x^2} = 0 \tag{3}$$

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

$$\hat{\beta}_1 = \frac{c_{XY}}{s_X^2} \tag{4}$$

$$\hat{\beta}_0 = \overline{y} - \hat{\beta}_1 \overline{x} \tag{5}$$

Unbiasedness The least-squares estimator is unbiased:

$$\mathbb{E}\left[\hat{\beta}_0\right] = \beta_0 \tag{6}$$

$$\mathbb{E}\left[\hat{\beta}_1\right] = \beta_1 \tag{7}$$

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

$$\mathrm{Var}\left[\hat{\beta}_1\right] = \frac{\sigma^2}{n s_X^2} \tag{8}$$

$$\mathrm{Var}\left[\hat{\beta}_0\right] = \frac{\sigma^2}{n}\left(1 + \frac{\overline{x}^2}{s_X^2}\right) \tag{9}$$

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.

2 In-Sample MSE vs. True MSE

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

$$(\beta_0, \beta_1) = \underset{(b_0, b_1)}{\mathrm{argmin}}\, \mathbb{E}\left[(Y - (b_0 + b_1 X))^2\right] \tag{10}$$

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

$$(\hat{\beta}_0, \hat{\beta}_1) = \underset{(b_0,b_1)}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} (y_i - (b_0 + b_1 x_i))^2 \tag{11}$$

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

$$\frac{1}{n} \sum_{i=1}^{n} (y_i - (b_0 + b_1 x_i))^2 \to \mathbb{E}\left[(Y - (b_0 + b_1 X))^2\right] \tag{12}$$

as $n \to \infty$. 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 2 shows a sequence of surfaces of the MSE as a function of (b_0, b_1). (The simulation code is in Figure 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.

2.1 Existence and Uniqueness

On any given finite data set, it is evident from Eqs. 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.

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

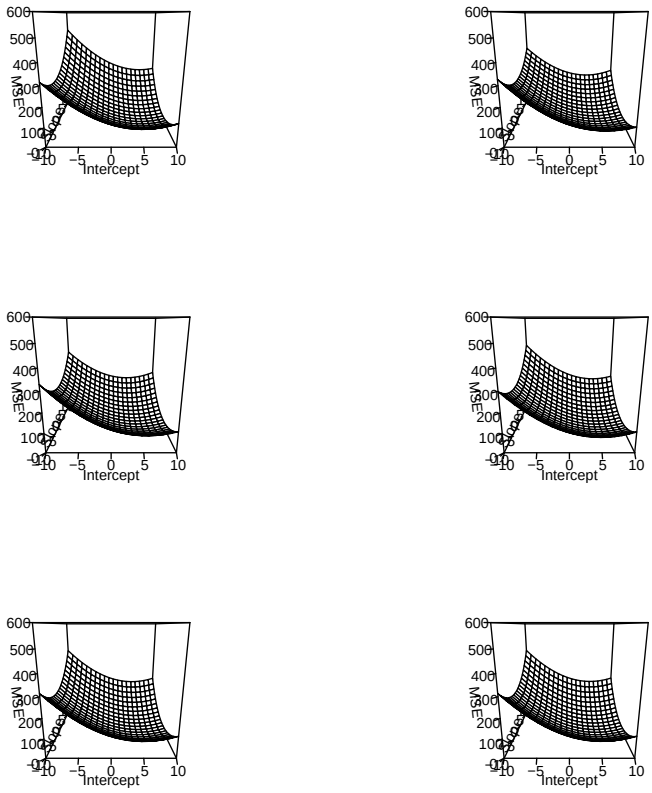
```
# Simulate from a linear model with uniform X and t-distributed noise
# Inputs: number of points; intercept; slope; width of uniform X distribution
  # (symmetric around 0); degrees of freedom for t
# Output: data frame with columns for X and Y
sim.linmod <- function(n, beta.0, beta.1, width, df) {
    # 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 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.*




```
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=1e5,beta.0=5,beta.1=-2,width=4,df=3))
in.sample.persp(sim.linmod(n=1e5,beta.0=5,beta.1=-2,width=4,df=3))
par(mfrow=c(1,1))
```

FIGURE 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.*

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 Lecture 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:

$$\hat{\beta}_1 = \frac{c_{XY}}{s_X^2} \tag{13}$$

$$= \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{s_X^2} \tag{14}$$

$$= \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})y_i - \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})\overline{y}}{s_X^2} \tag{15}$$

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:

$$\frac{1}{n}\sum_{i=1}^{n} z_i - \overline{z} = \frac{1}{n}\sum_{i=1}^{n} z_i - \frac{1}{n}\sum_{i=1}^{n}\overline{z} \tag{16}$$

$$= \overline{z} - \frac{n\overline{z}}{n} = 0 \tag{17}$$

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

$$\frac{1}{n}\sum_{i=1}^{n}(z_i - \overline{z})w = 0 \tag{18}$$

Thus

$$\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})\overline{y} = 0 \tag{19}$$

So

$$\hat{\beta}_1 = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})y_i}{s_X^2} \tag{20}$$

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

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i \tag{21}$$

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 $\overline{x}$:

$$y_i = \beta_0 + \beta_1\overline{x} + \beta_1(x_i - \overline{x}) + \epsilon_i \tag{22}$$

to substitute the above expression for y_i into Eq. 20:

$$\hat{\beta}_1 = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})(\beta_0 + \beta_1\overline{x} + \beta_1(x_i - \overline{x}) + \epsilon_i)}{s_X^2} \quad (23)$$

$$= \frac{\frac{\beta_0+\beta_1\overline{x}}{n}\sum_{i=1}^{n}(x_i - \overline{x}) + \frac{\beta_1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2 + \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})\epsilon_i}{s_X^2} \quad (24)$$

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. 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:

$$\hat{\beta}_1 = \beta_1 + \sum_{i=1}^{n}\frac{x_i - \overline{x}}{ns_X^2}\epsilon_i \quad (25)$$

Notice the form of Eq. 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:

$$\mathbb{E}\left[\hat{\beta}_1\right] = \mathbb{E}\left[\beta_1 + \sum_{i=1}^{n}\frac{x_i - \overline{x}}{ns_X^2}\epsilon_i\right] \quad (26)$$

$$= \beta_1 + \sum_{i=1}^{n}\frac{x_i - \overline{x}}{s_X^2}\mathbb{E}\left[\epsilon_i\right] = \beta_1 \quad (27)$$

or the variance:

$$\mathrm{Var}\left[\hat{\beta}_1\right] = \mathrm{Var}\left[\beta_1 + \sum_{i=1}^{n}\frac{x_i - \overline{x}}{ns_X^2}\epsilon_i\right] \quad (28)$$

$$= \mathrm{Var}\left[\sum_{i=1}^{n}\frac{x_i - \overline{x}}{ns_X^2}\epsilon_i\right] \quad (29)$$

$$= \sum_{i=1}^{n}\frac{(x_i - \overline{x})^2}{n^2 s_X^4}\mathrm{Var}\left[\epsilon_i\right] \quad (30)$$

$$= \sigma^2\frac{ns_X^2}{n^2 s_X^4} = \frac{\sigma^2}{ns_X^2} \quad (31)$$

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 9, and, for the ambitious, **?**), 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 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 Lecture 1 is the best prediction we can make. As x changes, 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

$$\hat{m}(x) = \hat{\beta}_0 + \hat{\beta}_1 x \tag{32}$$

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. 5,

$$\begin{aligned}
\hat{m}(x) &= \hat{\beta}_0 + \hat{\beta}_1 x & (33)\\
&= \overline{y} - \hat{\beta}_1 \overline{x} + \hat{\beta}_1 x & (34)\\
&= \overline{y} + (x - \overline{x})\hat{\beta}_1 & (35)
\end{aligned}$$

Using Eq. 25 and the definition of a sample mean,

$$\begin{aligned}
\hat{m}(x) &= \frac{1}{n}\sum_{i=1}^{n} y_i + (x - \overline{x})\left(\beta_1 + \sum_{i=1}^{n} \frac{x_i - \overline{x}}{n s_X^2}\epsilon_i\right) & (36)\\
&= \frac{1}{n}\sum_{i=1}^{n} (\beta_0 + \beta_1 x_i + \epsilon_i) + (x - \overline{x})\left(\beta_1 + \sum_{i=1}^{n} \frac{x_i - \overline{x}}{n s_X^2}\epsilon_i\right) & (37)\\
&= \beta_0 + \beta_1 \overline{x} + \frac{1}{n}\sum_{i=1}^{n} \epsilon_i + (x - \overline{x})\beta_1 + (x - \overline{x})\sum_{i=1}^{n} \frac{x_i - \overline{x}}{n s_X^2}\epsilon_i & (38)\\
&= \beta_0 + \beta_1 x + \frac{1}{n}\sum_{i=1}^{n} \left(1 + (x - \overline{x})\frac{x_i - \overline{x}}{s_X^2}\right)\epsilon_i & (39)
\end{aligned}$$

[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".

[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.

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:

$$\begin{aligned}
\mathbb{E}\left[\hat{m}(x)\right] &= \mathbb{E}\left[\beta_0 + \beta_1 x + \frac{1}{n}\sum_{i=1}^{n}\left(1 + (x-\overline{x})\frac{x_i - \overline{x}}{s_X^2}\right)\epsilon_i\right] && (40)\\
&= \beta_0 + \beta_1 x + \frac{1}{n}\sum_{i=1}^{n}\left(1 + (x-\overline{x})\frac{x_i - \overline{x}}{s_x^2}\right)\mathbb{E}\left[\epsilon_i\right] && (41)\\
&= \beta_0 + \beta_1 x && (42)
\end{aligned}$$

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?

$$\begin{aligned}
\mathrm{Var}\left[\hat{m}(x)\right] &= \mathrm{Var}\left[\beta_0 + \beta_1 x + \frac{1}{n}\sum_{i=1}^{n}\left(1 + (x-\overline{x})\frac{x_i - \overline{x}}{s_X^2}\right)\epsilon_i\right] && (43)\\
&= \mathrm{Var}\left[\frac{1}{n}\sum_{i=1}^{n}\left(1 + (x-\overline{x})\frac{x_i - \overline{x}}{s_X^2}\right)\epsilon_i\right] && (44)\\
&= \frac{1}{n^2}\sum_{i=1}^{n}\left(1 + (x-\overline{x})\frac{x_i - \overline{x}}{s_X^2}\right)^2 \mathrm{Var}\left[\epsilon_i\right] && (45)\\
&= \frac{\sigma^2}{n^2}\sum_{i=1}^{n} 1 + 2(x-\overline{x})\frac{x_i - \overline{x}}{s_X^2} + (x-\overline{x})^2\frac{(x_i - \overline{x})^2}{s_X^4} && (46)\\
&= \frac{\sigma^2}{n^2}\left(n + 0 + (x-\overline{x})^2\frac{n s_X^2}{n s_X^4}\right) && (47)\\
&= \frac{\sigma^2}{n}\left(1 + \frac{(x-\overline{x})^2}{s_X^2}\right) && (48)
\end{aligned}$$

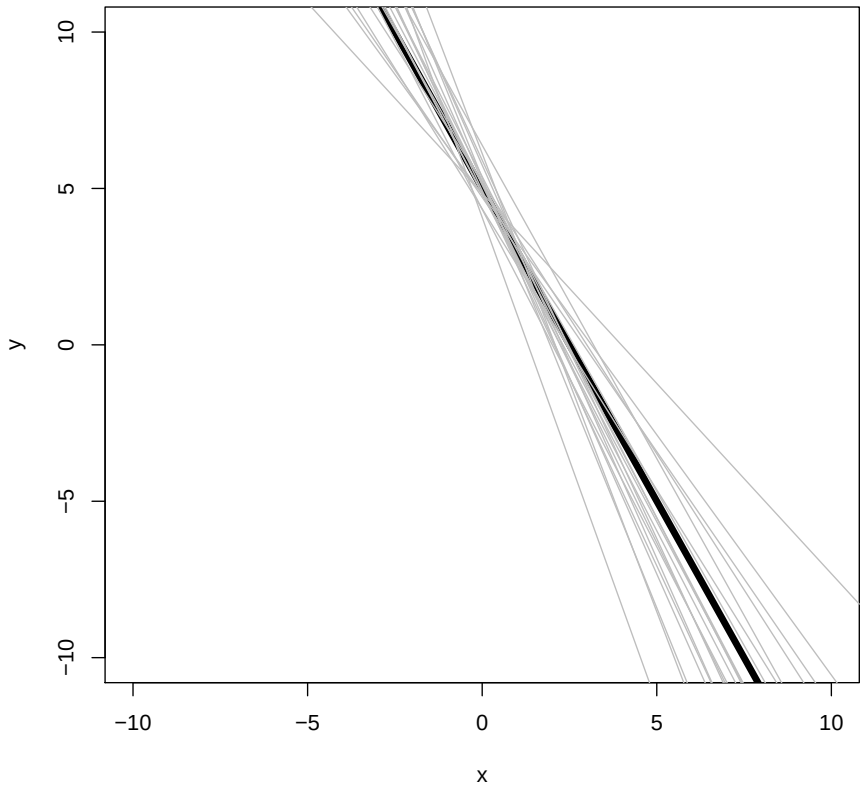
Notice what's going on here:

- 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. 48 is conditional on the x_i. If those are random, we need to use the law of total variance (as in the last lecture) to get the unconditional variance of $\hat{m}(x)$.

Figure 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 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* $X = 0$*).*

5 Estimating σ^2; Sum of Squared Errors

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

$$\mathbb{E}\left[(Y - (\beta_0 + \beta_1 X))^2\right] = \sigma^2 \tag{49}$$

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

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{m}(x_i))^2 \tag{50}$$

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)

$$s^2 = \frac{n}{n-2}\hat{\sigma}^2 \tag{51}$$

is an *un*-biased estimator of σ^2, though one with a larger variance. Some people, accordingly, use Eq. 51, rather than Eq. 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:

$$SSE = \sum_{i=1}^{n}(y_i - \hat{m}(x_i))^2 = \sum_{i=1}^{n}(y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i))^2 \tag{52}$$

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.

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 $\hat{m}(x_i)$:

$$e_i = y_i - \hat{m}(x_i) = y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i) \tag{53}$$

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

$$\epsilon_i = Y_i - (\beta_0 + \beta_1 x_i) \tag{54}$$

and it *is* similar, but it's *not* the same. The right-hand side of Eq. 54 involves the true parameters. The right-hand side of Eq. 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:

$$\frac{1}{n}\sum_{i=1}^{n} e_i(x_i - \overline{x}) = 0 \tag{55}$$

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

$$\frac{1}{n}\sum_{i=1}^{n} e_i = 0 \tag{56}$$

This is *reminiscent* of $\mathbb{E}[\epsilon] = 0$, but generally $n^{-1}\sum_{i=1}^{n}\epsilon_i \neq 0$. In fact, Eq. 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.

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 $r > 0$,

$$\mathbb{P}\left(|Z - \mu| \geq r\right) \leq \frac{\mathrm{Var}\left[Z\right]}{r^2} \tag{57}$$

In particular, we can say that

$$\mathbb{P}\left(|Z - \mu| \geq k\sqrt{\mathrm{Var}\left[Z\right]}\right) \leq \frac{1}{k^2} \tag{58}$$

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. 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].

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
##       4.857       -2.175
```

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
##    4.857223   -2.174934
```

`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
## 9.0800253 6.7603763 6.8558205 3.3152211 6.2856302 1.2357448 0.9678781
##         8         9        10
## 0.9695428 2.9113412 1.4778954
```

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

```
residuals(toy.lm)
```

```
##           1           2           3           4           5           6
##  0.33216288 -0.28195847 -0.16959613  0.56191598 -0.11683723 -0.65118228
##           7           8           9          10
##  0.60304673  1.01284494  0.04913496 -1.33953138
```

(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.

`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
##  2.6822890  0.5073551 -1.6675788 -3.8425127 -6.0174465
```

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

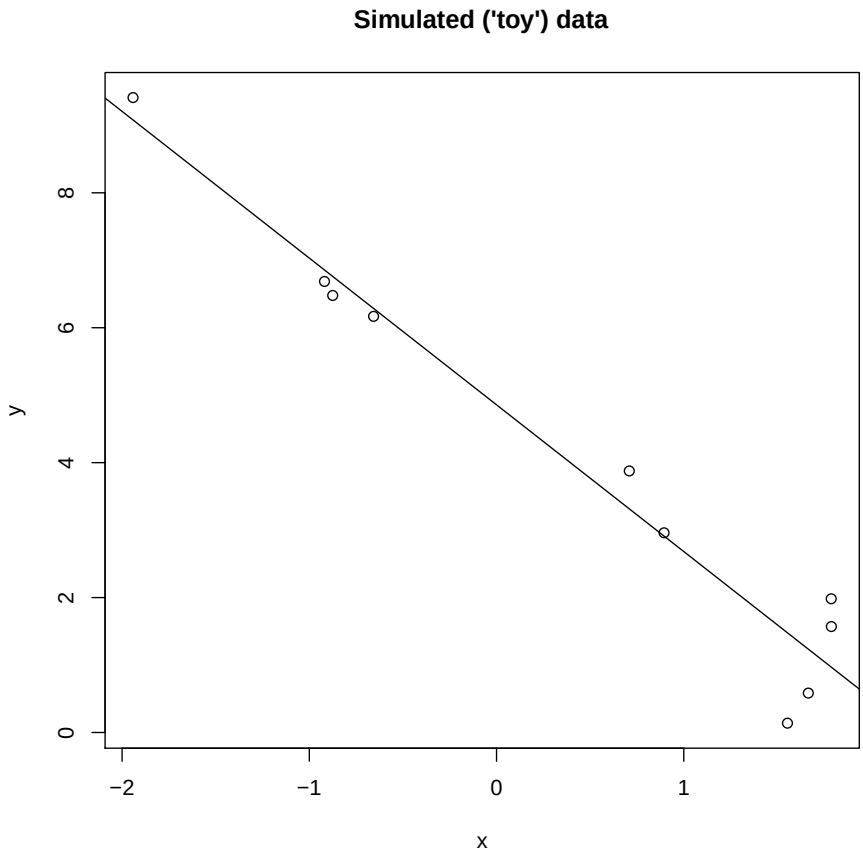
You might well think that if `newdata` were missing, then `predict` would throw an error. You might very well think that.

```
predict(toy.lm)
```

```
##         1         2         3         4         5         6         7
## 9.0800253 6.7603763 6.8558205 3.3152211 6.2856302 1.2357448 0.9678781
##         8         9        10
## 0.9695428 2.9113412 1.4778954
```

```
predict(toy.lm, data=data.frame(x=1:5))  # What's wrong here?
```

```
##         1         2         3         4         5         6         7
## 9.0800253 6.7603763 6.8558205 3.3152211 6.2856302 1.2357448 0.9678781
##         8         9        10
## 0.9695428 2.9113412 1.4778954
```




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

FIGURE 4: *Using* `abline` *to add the line of an estimated linear regression model to a plot.*

For reasons best known to the designers of R[7], when `newdata` is missing or mal-formed, `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(expr, envir, enclos):  object 'x' not found
```

Extraneous columns, however, are just ignored:

```
predict(toy.lm, newdata=data.frame(x=1:5, zebra=6:10))
```

```
##          1          2          3          4          5
##  2.6822890  0.5073551 -1.6675788 -3.8425127 -6.0174465
```

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
##  2.6822890  0.5073551 -1.6675788 -3.8425127 -6.0174465
##
## $se.fit
##         1         2         3         4         5
## 0.2508125 0.3602584 0.5074423 0.6678634 0.8339172
##
## $df
## [1] 8
##
## $residual.scale
## [1] 0.720964
```

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.)

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

An optional section, but a very useful one.

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

[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.

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.

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:

$$\widehat{\theta} = f(\widehat{\psi_1}, \widehat{\psi_2}, \ldots \widehat{\psi_p}) \tag{59}$$

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 $\widehat{\psi_i}$.

$$f(\psi_1^*, \psi_2^*, \ldots \psi_p^*) \approx f(\widehat{\psi_1}, \widehat{\psi_2}, \ldots \widehat{\psi_p}) + \sum_{i=1}^{p} (\psi_i^* - \widehat{\psi_i}) \left.\frac{\partial f}{\partial \psi_i}\right|_{\psi=\widehat{\psi}} \tag{60}$$

$$f(\widehat{\psi_1}, \widehat{\psi_2}, \ldots \widehat{\psi_p}) \approx f(\psi_1^*, \psi_2^*, \ldots \psi_p^*) + \sum_{i=1}^{p} (\widehat{\psi_i} - \psi_i^*) \left.\frac{\partial f}{\partial \psi_i}\right|_{\psi=\widehat{\psi}} \tag{61}$$

$$\hat{\theta} \approx \theta^* + \sum_{i=1}^{p} (\widehat{\psi_i} - \psi_i^*) f_i'(\widehat{\psi}) \tag{62}$$

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

$$\mathrm{Var}[a] = 0 \tag{63}$$

$$\mathrm{Var}[a + bX] = b^2 \mathrm{Var}[X] \tag{64}$$

$$\mathrm{Var}[a + bX + cY] = b^2 \mathrm{Var}[X] + c^2 \mathrm{Var}[Y] + 2bc\mathrm{Cov}[X, Y] \tag{65}$$

[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.

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. 62,

$$\mathrm{Var}\left[\widehat{\theta}\right] \approx \sum_{i=1}^{p} (f_i'(\widehat{\psi}))^2 \mathrm{Var}\left[\widehat{\psi}_i\right] + 2\sum_{i=1}^{p-1}\sum_{j=i+1}^{p} f_i'(\widehat{\psi}) f_j'(\widehat{\psi}) \mathrm{Cov}\left[\widehat{\psi}_i, \widehat{\psi}_j\right] \tag{66}$$

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

$$\mathrm{Var}\left[\widehat{\theta}\right] = \mathrm{Var}\left[\widehat{\psi_1}\right] + \mathrm{Var}\left[\widehat{\psi_2}\right] - 2\mathrm{Cov}\left[\widehat{\psi_1}, \widehat{\psi_2}\right] \tag{67}$$

just as we would find from the basic rules for arithmetic with variances. The approximation in Eq. 66 comes from the nonlinearities in f.

If the estimates of the initial quantities are uncorrelated, Eq. 66 simplifies to

$$\mathrm{Var}\left[\widehat{\theta}\right] \approx \sum_{i=1}^{p} (f_i'(\widehat{\psi}))^2 \mathrm{Var}\left[\widehat{\psi}_i\right] \tag{68}$$

and, again, the standard error of $\widehat{\theta}$ would be the square root of this. The special case of Eq. 68 is sometimes called *the* propagation of error formula, but I think it's better to use that name for the more general Eq. 66.

Exercises

To think through or practice on, not to hand in.

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

$$\mathrm{Var}[Y] + (\mathbb{E}[Y])^2 - 2b_0\mathbb{E}[Y] - 2b_1\mathrm{Cov}[X,Y] - 2b_1\mathbb{E}[X]\,\mathbb{E}[Y] + 2b_1\mathbb{E}[X] + b_1^2\mathrm{Var}[X] + b_1^2(\mathbb{E}[X])^2 \tag{69}$$

2. Show that if all $x_i = \overline{x}$, then the system of linear equations, Eqs. 2–3, 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, $\mathrm{Var}[\overline{y}] = \sigma^2/n$. You may treat the x_i as fixed in this calculation.

4. Derive Eqs. 6 and 9 from the results in §4. (Hint: $\hat{\beta}_0 = \hat{m}(x)$ for what value of x?) Is this a circular argument?

5. Prove Eq. 49.

6. Express the right-hand side of Eq. 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 $\frac{n-2}{n}\sigma^2$.