This week we'll be talking about various methods of computing functions of standard errors. We'll begin by discussing how to compute the standard error of a sum of two coefficients, followed by the more complex standard error of the prediction. Finally, we'll show how to compute nonlinear functions of coefficients.

Standard error of a linear combination of coefficients

Let's return to section 5 to think about wages. Suppose we run the following model:

$$\text{wage}_i = \alpha + \beta_1 \text{EDUC}_i + \beta_2 \text{TENURE}_i + \varepsilon_i$$

We can run this model in `R` as follows (after, of course, setting up our OLS function):

```
OLS <- function(y,X) {
  n <- nrow(X); k <- ncol(X)
  b <- solve(t(X) %*% X) %*% t(X) %*% y; names(b) <- "Estimate"
  e <- y - X %*% b; s2 <- t(e) %*% e / (n - k); XpXinv <- solve(t(X) %*% X)
  se <- sqrt(s2 * diag(XpXinv)); names(se) <- "Std. Error"
  t <- b / se; names(t) <- "t"
  p <- 2 * pt(-abs(t),n-k); names(t) <- "p"
  return(data.frame(b, se, t, p))
}

library(foreign)
library(xtable)
data <- read.dta("http://fmwww.bc.edu/ec-p/data/wooldridge/wage2.dta")
data <- data[ , c("wage", "educ", "tenure")]
data <- na.omit(data)
y <- data$wage
X <- cbind(1,data$educ,data$tenure)
OLS.out <- OLS(y,X)
xtable(OLS.out)
```

	b	se	t	p
1	53.52	79.56	0.67	0.50
2	61.15	5.64	10.84	0.00
3	11.18	2.44	4.58	0.00

Great!
But suppose the actual coefficients aren't what you actually care about. Maybe what you *really* want to compute is the additional wage benefit[1] of three years of education and two years at a company. Let's call it $d = 3b_1 + 2b_2$. Kind of weird, but easy enough:

```
(d <- 3 * OLS.out[2,1] + 2 * OLS.out[3,1])
```

```
[1] 205.7979
```

[1] Here I ask you to make the appropriate suspension of disbelief w.r.t the causal nature of this model.

However, this isn't really a meaningful answer without standard errors. How do we get those? If we were using `Stata`, the immediate answer is `lincom`. And in fact, there are canned functions in `R` that can do the same thing. But it's more interesting to do it by hand, and it gives us the opportunity to demonstrate three different ways of computing standard errors.

Analytical method

The first way is the analytical method. Here, it's fairly straightforward to compute the variance of d using the properties of the variance formula:

$$V(d) = V(3b_1 + 2b_2)$$
$$V(d) = 9V(b_1) + 4V(b_2) + (3 \times 2)\ \text{Cov}(b_1, b_2)$$

Fortunately, we can easily get estimates for $V(b_1)$, $V(b_2)$, and $\text{Cov}(b_1, b_2)$ by computing the variance-covariance matrix of the coefficients: $V(\mathbf{b}) = s^2(\mathbf{X}'X)^{-1}$. They will be the entries in (2,2), (3,3), and (2,3), respectively. To make this easier, we'll write a function that returns this matrix.

```
vcov <- function(y, X)  {
  n <- nrow(X); k <- ncol(X)
  b <- solve(t(X) %*% X) %*% t(X) %*% y
  e <- y - X %*% b
  s2 <- t(e) %*% e / (n - k)
  XpXinv <- solve(t(X) %*% X)
  vcov <- s2[1,1] * XpXinv
  return(vcov)
}
vcov <- vcov(y,X)
var.b1 <- vcov[2,2]
var.b2 <- vcov[3,3]
cov.b1b2 <- vcov[2,3]
(d.se <- sqrt(9 * var.b1 + 4 * var.b2 + 3*2*cov.b1b2))
```

```
[1] 17.69076
```

Great! We now know that our estimate of d is 205.8 with a standard error of 17.7. We can verify this in `R` using the `estimable()` command in the `gmodels` package (you may have to install it).

```
library(gmodels)
lm <- lm(data$wage ~ data$educ + data$tenure)
hm <- c(0,3,2)
estimable(lm, hm)
```

```
        Estimate Std. Error  t value  DF Pr(>|t|)
(0 3 2) 205.7979   17.77496 11.57797 932        0
```

Delta method

Let $\mathbf{A}(\boldsymbol{\beta}) \equiv \frac{\partial \mathbf{a}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}'}$. From Max's notes, we know the following:

$$\sqrt{N}(\mathbf{x}_N - \boldsymbol{\beta}) \xrightarrow{d} N(\mathbf{0}, \boldsymbol{\Sigma}) \Rightarrow \sqrt{\mathbf{N}}(\mathbf{a}(\mathbf{x_N}) - \mathbf{a}(\boldsymbol{\beta})) \xrightarrow{\mathbf{d}} \mathbf{N}(\mathbf{0}, \mathbf{A}(\boldsymbol{\beta})\boldsymbol{\Sigma}\mathbf{A}(\boldsymbol{\beta})')$$

This looks intimidating, but it's actually fairly straightforward to bend it to our setting. Let $\mathbf{x}_N \equiv \mathbf{b}$ and $\mathbf{a}(\mathbf{x}_N) \equiv d(\boldsymbol{\beta})$. This gives us that $\mathbf{A} \equiv \mathbf{D}(\boldsymbol{\beta}) = [0\ 3\ 2]$.

We know from the proof in section 4.2 of the lecture notes that $\sqrt{N}(\mathbf{b} - \boldsymbol{\beta}) \xrightarrow{d} N(\mathbf{0}, \sigma^{\mathbf{2}}(\mathbf{X}'\mathbf{X})^{-\mathbf{1}})$, so we immediately get that $V(d) \approx \mathbf{D}\sigma^2(\mathbf{X}'\mathbf{X})^{-1}\mathbf{D}'$. So to approximate the variance of our linear combination of coefficients, d, all we have to do is pre- and post-multiply $\sigma^2(\mathbf{X}'\mathbf{X})^{-1}$ by the gradient of d, $\mathbf{D}$. As usual, the standard error is just the square root of the variance. Putting it all together gives us our answer:

```
D <- matrix(c(0,3,2), ncol = 3)
(sqrt(D %*% vcov %*% t(D)))
```

```
         [,1]
[1,] 17.77496
```

Wow! It works! I'm as surprised as you are.

Bootstrapping the standard errors

Computing the standard error of the prediction

The standard error the prediction is an important quantity for econometricians who are interested in forecasting. It answers the question "how much variation should I expect to see when I use the coefficients from OLS to predict other values of y?" Modern statistical software has built-routines to calculate the prediction and the standard error of the prediction, but it's useful to know the calculation is performed. In fact, since we're using linear models, the prediction is just another linear combination!
Our model is

$$\mathbf{y} = \mathbf{Xb} + \boldsymbol{\varepsilon}$$

We want to compute $V(e^0) = V(\hat{y}^0 - y^0)$, the variance of the error in prediction for a particular set of covariates $\mathbf{x}^0$.

Analytical

First, we can calculate the variance analytically. This is similar in spirit to the analytical exercise above, but since we are working with more complex objects (coefficients, data, and disturbances), we'll use matrix notation. Still, everything that follows is just algebra and the use of the properties

of the variance function.

$$\begin{aligned} V(\hat{y}^0 - y^0) &= V(\mathbf{X}^0\mathbf{b} - \mathbf{X}^0\boldsymbol{\beta} - \boldsymbol{\varepsilon}^0) \\ &= \sigma^2 + V(\mathbf{X}^0(\mathbf{b} - \boldsymbol{\beta})) \\ &= \sigma^2 + V(\mathbf{X}^0\mathbf{b}) \\ &= \sigma^2 + \mathbf{X}^0 V(\mathbf{b})\mathbf{X}'^0 \\ &= \sigma^2 + \mathbf{X}^0\sigma^2(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'^0 \\ &= \sigma^2(1 + \mathbf{X}^0(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'^0) \end{aligned}$$

Of course, using the y^0 and $\mathbf{X}^0$ is only to make the point that we are actually computing the standard error of the prediction for a single set of data $\mathbf{X}^0$. Practically, we can tell `R` to compute the standard errors for all of observations all at once. We'll demonstrate using the `iris` dataset built into `R` and attempt to predict sepal length from sepal width and petal width[2].

```
iris.df <- iris
y <- iris.df$Sepal.Length
X <- cbind(1, iris.df$Sepal.Width, iris.df$Petal.Width)
b <- OLS(y,X)[ ,1]
head(pred.y <- X %*% b)
```

```
         [,1]
[1,] 5.048507
[2,] 4.848972
[3,] 4.928786
[4,] 4.888879
[5,] 5.088414
[6,] 5.402561
```

Getting the predictions for our data $\mathbf{X}$ is the easy part. Now we want to get some estimate of the standard error on these predictions. We'll do so using the formula we derived above:

```
e <- y - X %*% b
n <- nrow(X); k <- ncol(X)
s2 <- t(e) %*% e / (n - k)
XpXinv <- solve(t(X) %*% X)
pred.se <- diag(sqrt(s2[1,1] * (1 + X %*% XpXinv %*% t(X))))
head(pred.se)
```

```
[1] 0.4556677 0.4558185 0.4552114 0.4554239 0.4561838 0.4583894
```

To verify, we can use `lm()` and `predict()`:

```
lm <- lm(Sepal.Length ~ Sepal.Width + Petal.Width, data = iris.df)
predict.out <- predict(lm, se.fit = T)
all.equal(as.vector(predict.out$fit),as.vector(pred.y))
all.equal(as.vector(predict.out$se.fit),as.vector(pred.se))
```

[2] Dammit Jim, I'm a doctor, not a botanist!

```
[1] TRUE
[1] "Mean relative difference: 6.394198"
```

Something's not right. Let's investigate.

```
head(predict.out$se.fit)
head(pred.se)
```

```
[1] 0.06446854 0.06552559 0.06116029 0.06272227 0.06802035 0.08151061
[1] 0.4556677 0.4558185 0.4552114 0.4554239 0.4561838 0.4583894
```

Hm. That looks fishy. I'll spare you the suspense — `R` apparently has a different definition of the standard error of the prediction, and is returning the square root of $V(\hat{y}^0)$, rather than the square root of $V(\hat{y}^0 - y^0)$. This is why you should be careful with canned results! Anyway — we can replicate what they have easily:

```
pred.se.2 <- diag(sqrt(s2[1,1] * (X %*% XpXinv %*% t(X))))
all.equal(as.vector(predict.out$se.fit),as.vector(pred.se.2))
```

```
Warning message:
In sqrt(s2[1, 1] * (X %*% XpXinv %*% t(X))) : NaNs produced
[1] TRUE
```

Just pretend that warning message isn't there.

Delta method

We'll do the same thing we did earlier to use the Delta method. This time, $\mathbf{d}(\boldsymbol{\beta}) = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$, so $\mathbf{D}(\boldsymbol{\beta}) = [1 \ \mathbf{x}_1 \ x_2]$, where $\mathbf{x}_1$ is our vector of sepal widths and $\mathbf{x}_2$ is our vector of petal widths.

Non-linear functions of coefficients