ECO375 Tutorial 2
Monte Carlo Simulation / Interpretation Of OLS

Matt Tudball

University of Toronto Mississauga

September 21, 2017

Preliminaries: Random Number Generation

- We need to understand how random numbers are generated in a computer before we can discuss Monte Carlo simulations in any depth.
- Most random number generators use computational algorithms which produce large strings of seemingly random results, called pseudo-random numbers, from an initial value, called a seed. This is what Stata uses.
- Specifically, Stata uses an algorithm called the Mersenne Twister. In a computer, numbers are represented by 0's and 1's. This algorithm works by shifting those 0's and 1's around in a deterministic way which then appears random when they are represented as real numbers.
- You can set the seed to, for example, 375 in Stata by typing `set seed 375.`

Preliminaries: Data Generating Process

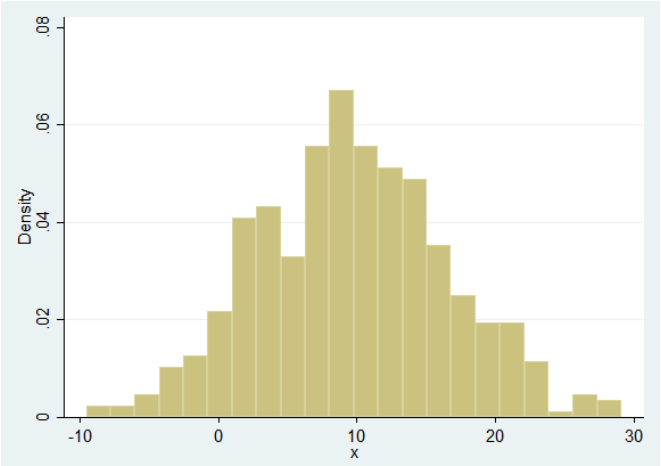
- The data generating process (DGP) is the true underlying mechanism that is generating the data we observe in our sample.
- Random number generators allow us to work in reverse of the usual statistical problem, by first specifying a DGP and then simulating data as though it were drawn randomly from it.
- When generating simulated data in Stata you must first specify how many observations Stata needs to make room for using the command `set obs #`.
- There are several distributions from which simulated data can be drawn in Stata. For example:
  - Uniform distribution on $(0, 1)$: `gen x = runiform()`
  - Normal distribution (μ, σ): `gen x = rnormal(`μ`,`σ`)`
  - Binomial distribution (n, p): `gen x = rnbinomial(n,p)`

Preliminaries: Example 1 (Drawing A Sample)

- Suppose we want to draw 500 observations from a normal distribution with mean 10 and standard deviation 7. Our code would look as follows:

```
set seed 375
set obs 500
gen x = rnormal(10,7)
```

- The histogram of this dataset looks like this:



- Note that your histogram looks exactly the same as mine since we both used the same seed.

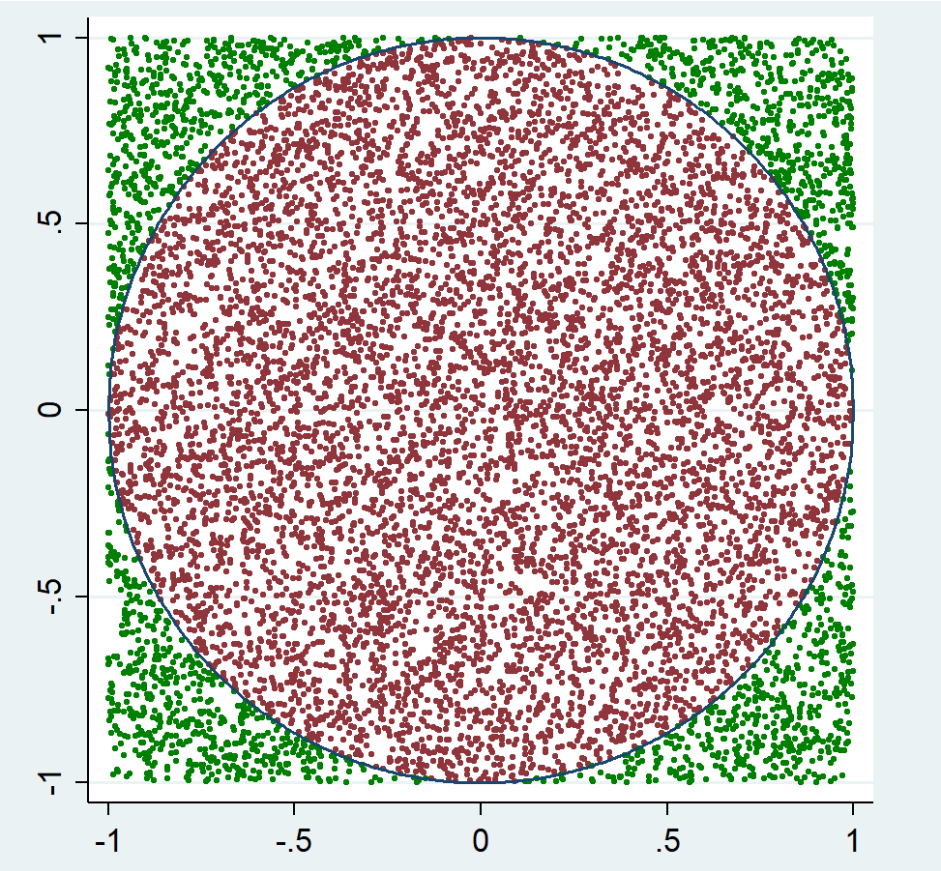
What Is A Monte Carlo Simulation?

- In general, a Monte Carlo simulation is a computer simulation that approximates solutions to numerical problems using large amounts of pseudo-randomly generated data from a given data generating process (this general form is seen in the Bonus Example).
- In statistical applications, it is common to simulate a large number of different samples, all of them with the same number of observations.
- This can be used to:
  - Explore the distributional properties of test statistics or estimators (as we will see in Example 2).
  - Evaluate the "closeness" of the exact sampling distribution to its asymptotic approximation, since the former is occasionally not known.

Bonus Example (Estimating The Value Of Pi)

- **Note:** This example is just for intuition. You will not be tested on it.
- Consider the unit square $[-1,1]x[-1,1]$ with a circle of radius $R = 1$ inscribed inside of it. We know that the area of the circle will be πR^2 and the area of the square will be $4R^2$. So the ratio of the area of the circle to the area of the square will be $\pi/4$.
- Now suppose we draw N random points from inside the unit square. We can do this by drawing the x-coordinate and y-coordinate separately from the unif(-1,1) distribution.
- We know that a point (x, y) is inside the unit circle if $x^2 + y^2 \leq 1$ and is outside of it otherwise.
- We can estimate $\pi/4$ by dividing the number of points that are drawn inside of the unit circle M, which is a random variable, by the total number of points drawn N.
- We therefore have an estimator $\hat{\pi} = 4M/N$.

Bonus Example (Estimating The Value Of Pi)



- In my simulation I get an estimate $\hat{\pi} = 3.28$ with $N = 100$, $\hat{\pi} = 3.204$ with $N = 1000$ and $\hat{\pi} = 3.164$ with $N = 10000$.

Example 2 (Sampling Distribution Of The Mean)

- Let's return to the example earlier of generating a random sample from a known distribution.
- Suppose we have a sample of 100 observations drawn from the unif(0,1) distribution and we are interested in the mean. We can type the following code:

```
set seed 1
set obs 100
gen x = runiform()
mean x
```

- You will get an estimate of 0.478.
- We could hypothetically draw another sample from the exact same distribution and get a different estimate.
- Clear your data and re-run the code with a seed of 2. You will get an estimate of 0.475 this time.

Example 2 (Sampling Distribution Of The Mean)

- You might begin to wonder what the distribution of these mean estimates will look like across many samples.
- This is the perfect set-up for a Monte Carlo simulation!
- Our simulation will proceed as follows:
  1. Draw 1,000 samples each with $N = 100$ observations from the unif(0,1) distribution.
  2. Calculate the mean of each of these samples.
  3. Evaluate the mean and standard error of these sample means. Plot the distribution of the means in a histogram. This will be the so-called sampling distribution.

Example 2 (Sampling Distribution Of The Mean)

- There are a couple of additional commands we need to learn in order to do this.
- First we need to tell Stata where we want to store the means from our 1,000 random samples. We can do this using the command `postfile`.
- Begin your code by typing:
  `postfile buffer xhat using "$path/ex2_100.dta"`
- This will tell Stata that you want to save your means as a variable xhat in a file called ex2_100.dta.
- Once you have calculated a sample mean using the command `mean` as above, you can save it as a new observation in ex2_100.dta by typing `post buffer (_b[x])`.
- Once you have finished adding observations to ex2_100.dta you can save the file by typing `postclose buffer`

Example 2 (Sampling Distribution Of The Mean)

- Second we need to tell Stata that we want to loop over 1,000 random samples. We can do this efficiently using the command `forvalues`.
- Suppose, for example, we want to count to 10. We can type the following:

```
forvalues i = 1/10 {
    display `i'
}
```

- This will loop from 1 to 10 and display the current iteration at each step. Within the loop the current iteration is identified by the placeholder `i`.
- In our Monte Carlo simulation we will loop from 1 to 1,000 and generate a new random sample in each iteration, saving its mean as a new observation in the file ex2_100.dta.

Example 2 (Sampling Distribution Of The Mean)

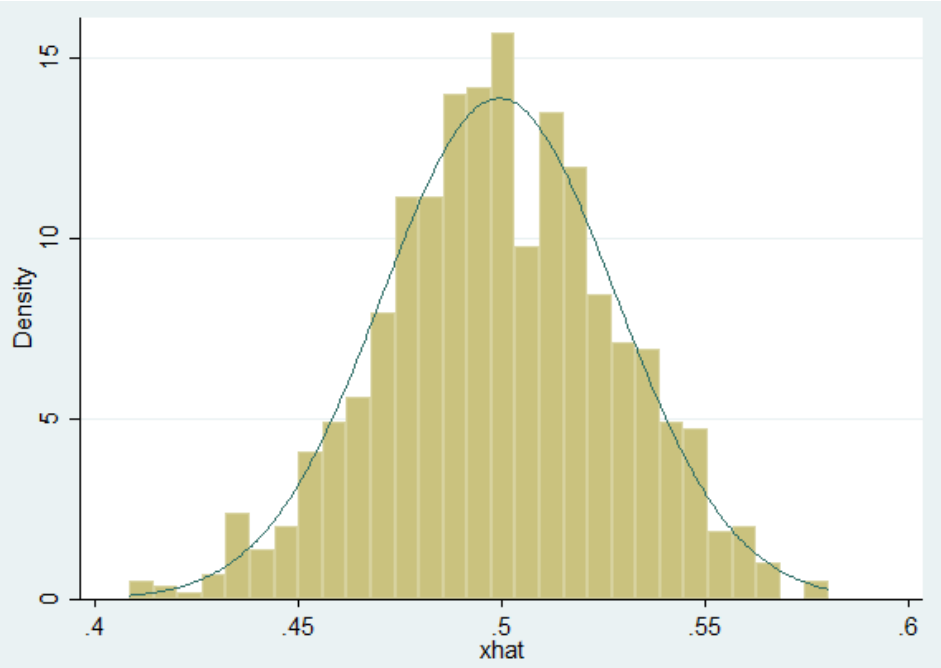
- Let's now put the code together:

```
postfile buffer xhat using "$path/ex2_100.dta"
forvalues i = 1/1000 {
    qui set seed `i'
    qui drop _all
    qui set obs 100
    qui generate x = runiform()
    qui mean x
    post buffer (_b[x])
}
postclose buffer
use "$path/ex2_100.dta", clear
```

- We can suppress the terminal output in Stata using the command `qui`.

Example 2 (Sampling Distribution Of The Mean)

- Now that we have our dataset of sample means loaded we can begin to explore it.
- You should be able to see that the mean of xhat $\hat{\mu}_M = 0.499$, which very close to the expected value $\mu = 0.5$.
- You can also see that the standard error of xhat $\hat{\sigma}_M = 0.0278$, which is very close to $\sigma/\sqrt{100} = 0.0289$ where $\sigma = 1/\sqrt{12}$ is the standard deviation of unif(0,1).



In-Class Exercise

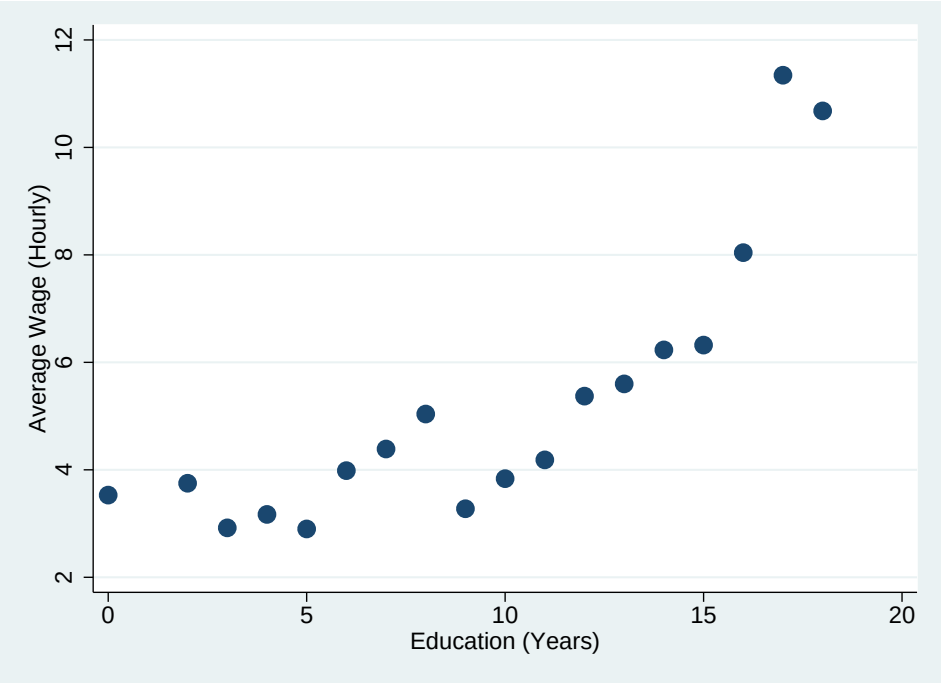
- Carry out a new simulation identical to the one in Example 2 except generating $N = 1000$ observations per sample.
- Compare the mean and standard error of xhat calculated in this new simulation with the ones calculated in the last simulation with $N = 100$. Does the mean look different? Does the standard error look different?
- If the standard errors look different, how do they compare to one another? How does this relate to the difference in sample sizes?

Interpretation Of OLS With Non-Linearities

- Recall that OLS assumes a linear relationship $y_i = \beta_0 + \beta_1 x_i + u_i$.
- It is not, however, always reasonable to assume a linear relationship in our data. Fortunately, many non-linear relationships can be made approximately linear with an appropriate transformation of the data.
- Our interpretation of the regression coefficients often changes when we apply these transformations.

Log-Linear Regression

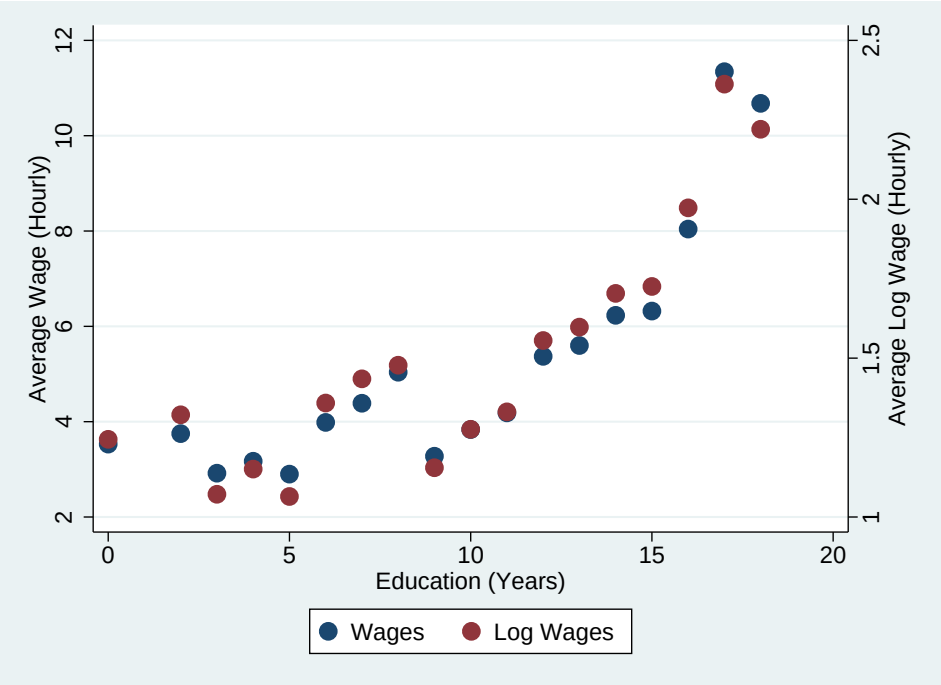
- In the WAGE1.dta dataset we can plot the average hourly wage against years of education.



- Notice that it appears as though $wage = e^{\beta_0 + \beta_1 educ + u}$.
- We can take the natural log of both sides to obtain $ln(wage) = \beta_0 + \beta_1 educ + u$.

Log-Linear Regression

- We can generate a new figure plotting average log wages against years of education.



- You can see that the log transformation flattens the relationship at the smallest and largest values of *educ*, making it more consistent with the SLR.1 linearity assumption.

Interpretation Of Log-Linear Regression

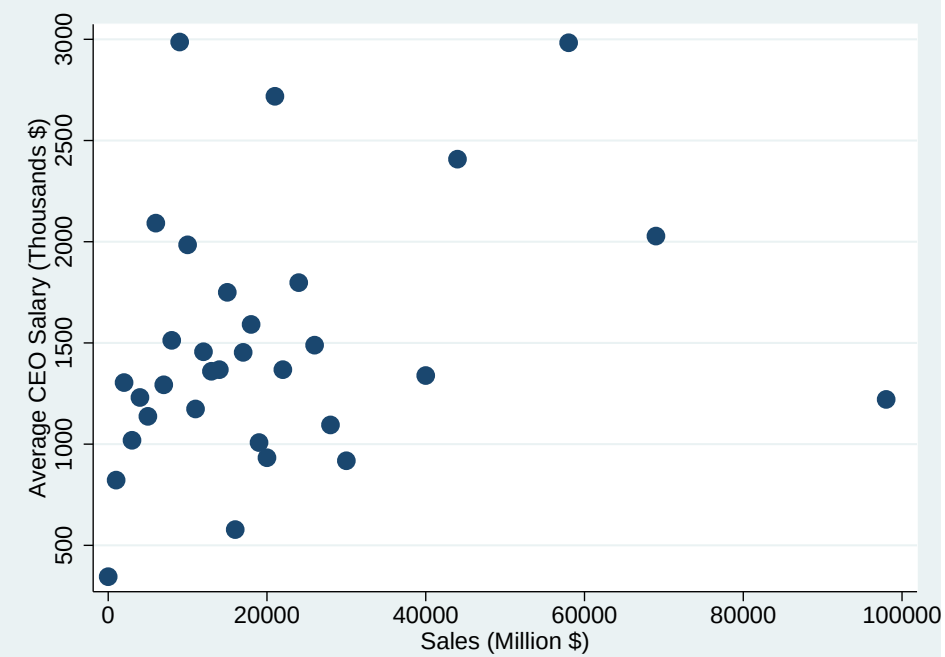
- You might be wondering, how do we interpret β_1 in this transformed regression $ln(y) = \beta_0 + \beta_1 x + u$?
- Note that in general $\beta_1 = \frac{\partial y}{\partial x}$.
- In the log-linear regression, $\beta_1 = \frac{\partial ln(y)}{\partial x} = \frac{\partial ln(y)}{\partial y}\frac{\partial y}{\partial x} = \frac{1}{y}\frac{\partial y}{\partial x}$.
- It may be more concrete to think of a discrete change $\Delta x = 1$. Then $\beta_1 = \frac{1}{y}\frac{\Delta y}{\Delta x} = \frac{\Delta y}{y} = (\%\text{ change in } y)/100$.
- In plain words, we interpret β_1 by saying on average "a 1 unit increase in x is associated with a $100\beta_1\%$ increase in y."

Example 3 (Log-Linear Regression)

- Suppose we are interested in the relationship between hourly wages and years of education. Let's begin by loading the dataset WAGE1.dta which can be found on my website at matthewtudball.com.
- First generate the natural log of each person's wages `gen ln_wage = ln(wage)`.
- Now we can run a log-linear regression by `reg ln_wage educ`.
- You will obtain estimates $\hat{\beta}_0 = 0.584$ and $\hat{\beta}_1 = 0.083$.
- As we learned above, we would interpret $\hat{\beta}_1$ by saying that on average a 1 year increase in education is associated with an 8.3% increase in hourly wages.
- $\hat{\beta}_0$ does not have a very meaningful interpretation as it is simply the average log wage when $educ = 0$.

Log-Log Regression

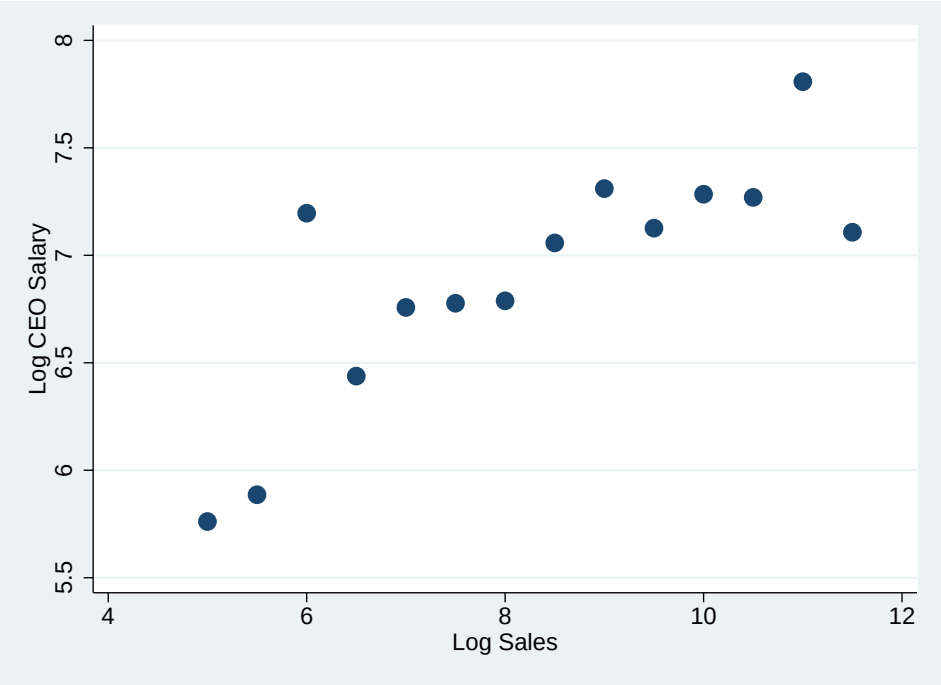
- In the CEOSAL1.dta dataset we can plot the average CEO salary against the amount of their company's sales (in groups of millions \$).



- A log-log model is useful when we think the relationship is a constant elasticity model $salary = e^{\beta_0} sales^{\beta_1} e^{u}$.
- When we take the natural log of both sides we obtain $ln(salary) = \beta_0 + \beta_1 ln(sales) + u$.

Log-Log Regression

- We can generate a new figure plotting average log CEO salary against average log sales.



- As before you can see that this transformation has led to a much more linear relationship which is more consistent with the SLR.1 linearity assumption.

Interpretation Of Log-Log Regression

- You might be wondering, how do we interpret β_1 in this transformed regression $ln(y) = \beta_0 + \beta_1 ln(x) + u$?
- In the log-log regression $\beta_1 = \frac{\partial ln(y)}{\partial ln(x)} = \frac{\partial ln(y)}{\partial y}\frac{\partial y}{\partial ln(x)} = \frac{1}{y}\frac{\partial y}{\partial ln(x)}$
- Let's again consider a discrete change $\Delta ln(x) = 1$, then $\beta_1 = \frac{1}{y}\frac{\Delta y}{\Delta ln(x)}$
- We need to recall two basic math facts:
  - $ln(1+\epsilon) \approx \epsilon$ for ϵ small.
  - $\frac{x_2}{x_1} = 1 + \frac{x_2 - x_1}{x_1} = 1 + (\%\text{ change in } x)/100$
- Then $\Delta ln(x) = ln(x_2) - ln(x_1) = ln(\frac{x_2}{x_1}) = ln(1 + \frac{x_2 - x_1}{x_1}) \approx \frac{x_2 - x_1}{x_1} = (\%\text{ change in } x)/100$.
- Putting this together, $\beta_1 = \frac{\%\text{ change in } y}{\%\text{ change in } x}$
- So in plain words, we interpret β_1 by saying on average "a 1% increase in x is associated with a β_1% increase in y".

Example 4 (Log-Log Regression)

- Suppose we are interested in the relationship between CEO salaries and their company's sales. Let's begin by loading the dataset CEOSAL1.dta which can be found on my website at matthewtudball.com.
- First generate the natural log of CEO salaries `gen ln_salary = ln(salary)`.
  Do the same for sales `gen ln_sales = ln(sales)`.
- Now run the log-log regression by `reg ln_salary ln_sales`.
- You will obtain estimates $\hat{\beta}_1 = 0.257$ and $\hat{\beta}_0 = 4.822$.
- We can interpret $\hat{\beta}_1$ by saying that on average a 1% increase in a company's sales is associated with a 0.257% increase in their CEO's salary.

Binary Independent Variable

- Suppose that x is binary (i.e. it only takes on values 0 or 1). How would be interpret a linear regression $y = \beta_0 + \beta_1 x + u$?
- Note that $E(y|x = 1) = \beta_0 + \beta_1$ and $E(y|x = 0) = \beta_0$.
- Thus $\beta_1 = E(y|x = 1) - E(y|x = 0)$ and $\beta_0 = E(y|x = 0)$.
- Interpretation of this regression is therefore very simple. Suppose that x is completing a university degree and y is hourly wages.
- We can say that completing a university degree corresponds to a β_1 increase in hourly wages on average.
- We can also say that β_0 is average hourly wages among those without a university degree.