

Lab 13

Constructing Confidence Intervals

In this lab, we will construct and visualize confidence intervals for different sample sizes and different levels of confidence.

Sampling Data

Please load the following dataset into *Stata*.

```
. use http://www.stat.ucla.edu/labs/datasets/cilab.dta
```

We have thirty variables regarding stocks listed in the Standard and Poor's 500. We will focus on the percentage return during a twenty-six week period (*pctchg26wks*).

```
. histogram pctchg26wks, norm  
. summarize pctchg26wks
```

Looking at the statistical summary of this variable, we can see that the average stock value in the Standard and Poor's 500 dropped 12.6% during this 26 week period, with a maximum loss of 92% to a maximum gain of 99%. The histogram shows us that the population distribution is approximately normal.

We are going to use this data set and this variable *pctchg26wks* to test the theory of confidence intervals. In general, the way we construct a confidence interval is by obtaining a sample and then constructing an interval around the mean of that sample. This interval is intended to give us an estimate of the true population mean, which in general is unknown. We are going to use a sort of reverse psychology. *We know* the true population mean for the percentage return for the S & P's 500 stocks over this particular 26 week period and *we know* the true population standard deviation (23.6) of this variable. The dataset we loaded contains the entire population! We can test the theory of confidence intervals by taking samples of the dataset and constructing confidence intervals around the means of those samples. Then we can check and see if these confidence intervals, that we estimated, contain the true mean or not.

We will use a statistical technique called *bootstrapping*. This simply means we will take simple random samples of the dataset with replacement. For the purposes of constructing confidence intervals, we are not really interested in the sample itself, we are more interested in the mean of the sample. The **bootstrap** command will bypass the output of individual samples and give us the sample mean for each sample it draws. The **bootstrap** command takes as many samples as you want of the specified variable and then takes the mean of each of those samples. It outputs a new dataset, where each observation represents a single sample from the original population.

***** Important *****

This new file should be saved into your home directory. To ensure this occurs, change your working directory by clicking on the File menu and selecting "Set Working Folder." The default is your "Documents" folder. This is the correct location, so click on the "Choose" button. Now any files you create will show up in your documents folder.

Now to ensure we truly are taking random samples, we want to randomly set a seed for *Stata* to start from.

```
. set seed YOUR STUDENT ID NUMBER
```

Finally, we are ready to start sampling.

```
. bootstrap "summarize pctchg26wks" "r(mean)", reps(100)
size(16) dots saving (cidata) replace
```

This command will take 100 random samples of size 16 and calculate the mean of the variable *pctchg26wks* for each sample. It will then save this information in a file called *cidata.dta*.

Now we want to open up this dataset we just created and construct a confidence interval for each of the 100 samples.

```
. use cidata, clear
```

When we issue the command

```
. list
```

we see one variable *_bs_1* (short for bootstrap one), which is a list of 100 means from the 100 samples of size 16 that we selected from the S&P's 500 stocks. We can look at the distribution of these sample means. (Remember the Central Limit Theorem tells us that as the size of our samples increase, the distribution of the sample means becomes more and more normally distributed.)

```
. histogram _bs_1, norm
```

Question 1: How does this distribution compare to the distribution of pctchg26wks?

Constructing & Visualizing Confidence Intervals

For each sample mean, we want to generate an appropriate confidence interval. Recall the formula for constructing a confidence interval when the standard deviation σ is known.

$$\bar{x} \pm z^* \frac{\sigma}{\sqrt{n}}$$

We know σ (the true population standard deviation) is 23.66127. We also know that n (the size of the sample) is 16. We want to explore what happens to confidence intervals when we change our level of confidence.

We will create 68%, 85%, and 93% confidence intervals for each of our 100 samples. The technique we will use is to create the lower bound of the confidence interval ($\bar{x} - z^* \frac{\sigma}{\sqrt{n}}$) and then the upper bound of the confidence interval ($\bar{x} + z^* \frac{\sigma}{\sqrt{n}}$).

For a 68% confidence interval, $z^* = 1.00$. The variable `_bs_1` contains all our sample means (\bar{x}).

```
. generate lower68 = _bs_1 - 1.00*23.66127/sqrt(16)
. generate upper68 = _bs_1 + 1.00*23.66127/sqrt(16)
```

Note: If you make a mistake when generating your new variables, use the `drop` command to remove variables from your dataset and reissue the correct command. For example, if I messed up and typed `generate upper68 = _bs_1 + 1.00*23.66127/sqrt(160)`, then I could remove this variable by typing `drop upper68` and then regenerate the variable correctly.

```
. list
```

Look at the data. As you can see, the variables `lower68` and `upper68` form an interval surrounding the `_bs_1` variable.

Question 2: We know that the true mean for this population is -12.6%. According to the theory of confidence intervals, how many of these confidence intervals should contain the true mean?

We can actually determine exactly how many of our confidence intervals contain the true mean.

```
. count if lower68 <= -12.567 & upper68 >= -12.567
```

This command counts the value only if the lower bound is below the true population mean **and** the upper bound is above the true population mean.

Question 3: How many of your 68% confidence intervals captured the true population mean? Does this number surprise you or does it seem about right?

We can visualize the confidence intervals by plotting them side by side. To do this we must create a helper variable...

```
. gen num = _n
```

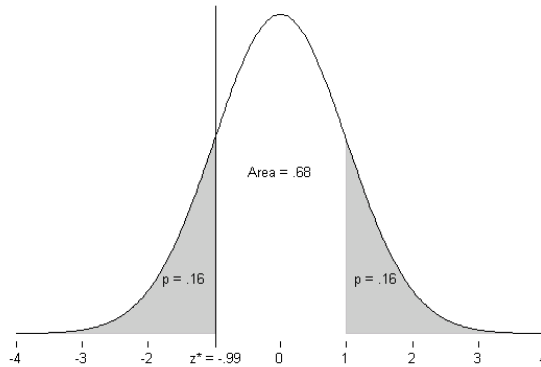
Then issue the following graph command.

```
. twoway (rcap upper68 lower68 num) (scatter _bs_1 num,  
msymbol(o)), yline(-12.567) title("100 68% confidence intervals  
from samples n=16")
```

Each of your confidence intervals pop up in the resulting graph. Each line corresponds to exactly one confidence interval and the dot in the middle corresponds to the \bar{x} for that confidence interval. We inserted a line going through the confidence intervals at the true population mean of -12.567 . As you can see, some of the confidence intervals capture this true mean and others don't. This is the caveat of confidence intervals. In a real life setting, we have no way of knowing if the one sample we have is one of the cases that does not capture the true mean! This is why large sample sizes and high levels of confidence are so important.

If you want, you can make this graph even fancier. Wouldn't it be nice if the graph somehow highlighted the confidence intervals that did not capture the true population mean?

```
. twoway (rcap upper68 lower68 num) (rcap upper68 lower68 num  
if upper68 < -12.567 | lower68 > -12.567, blcolor(red))  
(scatter _bs_1 num, msymbol(o)), yline(-12.567) title("100 68%  
confidence intervals from samples n=16")
```



This basically colors the interval red if the upper bound is below -12.567 or the lower bound is above -12.567 .

Question 4: What is the length of a 68% confidence interval in this setting?

Next we repeat the process by generating 85% and then 93% confidence intervals. For the 68% confidence interval, we told you z^* , if we hadn't, you could have looked it up on the standard normal table or remembered the 68-95-99.7 rule. *Stata* gives us another option. Using the `invnorm` function, if we give *Stata* a probability p , it will return the z -value such that the area below it will equal p . For example, suppose you had not been given the z^* value for a 68% confidence interval and you didn't know the 68-95-99.7 rule. To determine the z^* value, first we must divide $1 - .68$ by 2 to determine the correct probability we want to look up, and then we input that value into the `invnorm` function.

```
. di (1 - .68)/2
. di invnorm(.16)
```

As you can see, *Stata* returns a value of $-.9945$ or rounded even further -1 . We, of course, use the positive value for z^* .

Stata Note: `di` is the *Stata* equivalent to a calculator; it is short for `display`.

Question 5: What is the z^ value for an 85% confidence interval?*

Question 6: What number will you be adding and subtracting to “_bs_1” to obtain a 85% confidence interval? Based on this, what will be the length of a 85% confidence interval?

Construct variables *lower85* and *upper85* similarly to the way you constructed *lower68* and *upper68*. Be sure you make the appropriate change in the formula.

Question 7: According to theory, how many of your 85% confidence intervals are expected to capture the true population mean? How many of your 85% confidence intervals actually capture the true population mean?

Create a graph, with the appropriate title, of your 85% confidence intervals.

Repeat the process for 93% confidence intervals.

Question 8: What is the z^ value for a 93% confidence interval?*

Question 9: According to theory, how many of your 93% confidence intervals are expected to capture the true population mean? How many of your 93% confidence intervals actually capture the true population mean?

Supplementary Section on Do-files

This lab is a perfect example for when *Stata* do-files come in handy. The assignment for this lab is essentially a replication of the in-class portion of the lab. The main difference is you need to change the sample size n . A do-file is a file that contains one *Stata* command per line. When run, it executes each of the commands.

You can type commands directly into a file, save it with the `.do` extension and then do it in *Stata*, but in our case there is an even easier way.

Hold down the Control button on the keyboard and click on the Review window. You now have two options. First, “Copy Review Contents to Do-file Editor”. This will open the do-file editor window with the contents of the review window pasted in. Second, “Save Review Contents...”. This method allows you to save the file as a do-file. For the purpose of this lab, click on “Save Review Contents...” and save the file as `cilab.do`. Now type the following command.

```
. doedit cilab
```

This will bring up a text window with the commands you entered. A few corrections need to be made before this file will run. Add the option `clear` to the `use` command. In other words, the first line of your do-file should look like:

```
use http://www.stat.ucla.edu/labs/datasets/cilab.dta, clear
```

This is needed if data is already in *Stata* memory. The do-file won't run without it. Once you have added that option, re-save the do-file.

The next step is more user dependent. You need to clean up any potential errors you made when working through the in-class portion of the lab. Try doing the do-file.

```
. do cilab
```

What happens? Does it run all the way through until you see `end of do-file` in green? Or do you get a red error message? If you encounter a red error message, you need to find the error in the do-file and remove it.

Basically, your do-file should consist of one correct command per line. Clean up your do-file until it runs through.

Now to do the assignment, you need to change the sample size. First, we look at the same confidence intervals for sample size 36. Instead of losing the original do-file, save the do-file under a new name, such as `cilab36.do`. Now, change the appropriate commands in the do-file and maybe delete some unnecessary commands, such as the `list` commands.

You can now run this do-file, but you may notice a problem. The graphs fly by! If you want to print or save any of the graphs, this may be a problem. You can avoid this by doing only part of your do-file at a time. Highlight the portion you want to *do*. Then click on the “Do current file” button in the do-file editor window. Only the portion highlighted will be processed.



Assignment

What happens to the length of confidence intervals if we change the sample size n ? What happens to the length of confidence intervals as we change the level of confidence? What happens to the distribution of `_bs_1` as we change the sample size n ?

Reload the original dataset and run a new bootstrap command, this time with samples of size 36. (You do not need to reset the seed, this only needs to be done once.)

Question 10: How many of your confidence intervals “catch” the true population mean for each of the three levels of confidence when $n = 36$?

Question 11: Are corresponding confidence intervals using samples of size 36 longer or shorter than those for samples of size 16?

What happens if we use samples of size 100? Reload the original dataset and run a new bootstrap command using samples of size 100.

Question 12: How many of your confidence intervals “catch” the true population mean for each of the three levels of confidence when $n = 100$?

Question 13: Are corresponding confidence intervals using samples of size 100 longer or shorter than those for samples of size 36?

Question 14: As we increase the confidence level from 68% to 85% to 93%, what happens to the length of our intervals?

Question 15: What happens to the distribution of \bar{b}_s as the sample size increases from 16 to 36 to 100? Discuss the changes that occur to the mean and the spread.