Empirical exercise – Inference – Confidence interval

1. Aim of the exercise

You want to analyse the factors that determine the test scores of students in high school. You collect data on test scores, student teacher ratio, and percentage of English learners in a given class, among a set of other characteristics. Using this data you estimate a linear regression model where the explanatory variables are the student teacher ratio and the percentage of English learners in the class, and the outcome variable is the test scores. You are especially interested in the effect of student teacher ratio. You want a formal measure of how precisely this coefficient is estimated. There are different measures that serve to this purpose. One of the main measures is the confidence interval.

What underlies the rationale of a confidence interval is repeated sampling from the population. In practice we are not able to take samples from the population repeatedly. This makes it difficult to understand the rationale of a confidence interval and hence to interpret it. However, we can act as if we are able to take samples from the population repeatedly in a simulation exercise. In this exercise we will do this which will help to (better) understand the meaning of a confidence interval.

2. Empirical exercise on CI

2.1. Load the data

Load the data in .mat format into MATLAB.

```
clear;
load 'M:\exercise.mat';
clearvars -except testscr str el_pct;
```

2.2. Create the systematic component of the regression equation

Create the systematic component of the regression equation.

```
y = testscr;
N_obs = size(y,1);
x_0 = ones(N_obs,1);
X = [x_0 str el_pct];
```

2.3. Obtain the OLS coefficient estimate and the standard error estimate of it

You decide to use the OLS estimator to estimate the parameters of the regression equation. Obtain the OLS coefficient estimate of the independent variable of interest, which is `str`, and

the standard error estimate of it, using the supplied MATLAB function file that calculates OLS statistics.

```
LSS = exercisefunction(y,X);
B_hat_k = LSS.B_hat(2,1); % k = str.
B_hat_k_SEE = LSS.B_hat_SEE(2,1);
```

2.4. Carry out a t test on the population coefficient

Test whether the population coefficient of student teacher ratio is different from a hypothesised value of zero. Is this a one-tailed or two-sided t test? Calculate the critical value of the test using the degrees of freedom of the distribution of the t statistic, and assuming a significance level of 0.05. Compare the t value with the critical t value. What do you conclude?

```
t = B_hat_k/B_hat_k_SEE;
t_df = N_obs-size(X,2);
t_c = tinv(0.975,t_df);
```

2.5. Construct the CI for the population coefficient

Calculate the confidence interval for the population coefficient of the independent variable student teacher ratio. How do you interpret this confidence interval?

Assume that the OLS estimator $\hat{\beta}_k$ follows a normal distribution with mean β_k and variance $\sigma^2_{\hat{\beta}_k}$. That is, $\hat{\beta}_k \sim N\left[\beta_k, \sigma^2_{\hat{\beta}_k}\right]$. We can standardise $\hat{\beta}_k$ so that it has a standard normal distribution: $\hat{\beta}_k - \beta_k/\sigma_{\hat{\beta}_k} \sim N[0,1]$. $\sigma_{\hat{\beta}_k}$ is an unobserved population parameter. Replace it with its unbiased estimator $s_{\hat{\beta}_k}$. Then, $\hat{\beta}_k - \beta_k/s_{\hat{\beta}_k} \sim t[n-K]$. For $n-K = 417$, we can then state that $\text{Prob}\left(-1.9657 < \hat{\beta}_k - \beta_k/s_{\hat{\beta}_k} < 1.9657\right) = 0.95$. The interpretation of this expression is that the probability that the random variable $\hat{\beta}_k - \beta_k/s_{\hat{\beta}_k}$ is between the stated boundaries is 95 percent. The stated probability is equivalent to $\text{Prob}\left(\hat{\beta}_k - 1.9657 s_{\hat{\beta}_k} < \beta_k < \hat{\beta}_k + 1.9657 s_{\hat{\beta}_k}\right) = 0.95$. At this instance the interpretation changes. The interpretation is for the unique nonrandom population parameter β_k. The end points of the interval $\left[\hat{\beta}_k - 1.9657 s_{\hat{\beta}_k}, \hat{\beta}_k + 1.9657 s_{\hat{\beta}_k}\right]$ are random! The interval does not take one value but a different value from one sample to another. Suppose that we randomly collect an infinite number of samples (repeated sampling, or sampling in the long run), and construct a particular interval using each sample. The stated probability tells that 95 percent of these intervals will include β_k. This is where the probabilistic interpretation comes from. Given the single sample data at hand, we could calculate only one interval estimate which is $[-1.1013 \pm 1.9657 * 0.3803] = [-1.8488, -0.3530]$. Once we construct this interval using the sample data at hand, the end points of the interval are not random anymore! Therefore, the probability that β_k is in this interval is either 0 or 1. Hence, it is incorrect to say that the probability that β_k is in $[-1.8488, -0.3530]$ is 95 percent. The interval we computed is just an estimate of one of those intervals that contain β_k 95 percent of the times. The correct interpretation is the following: In repeated sampling, the probability that intervals like $[-1.8488, -0.3530]$ will contain the true β_k is 95 percent. The probability

that this particular non-random interval includes the true β_k is either 0 or 1.

Confidence interval is also called the 'interval estimate' because it provides a range of the possible estimates of the population coefficient, whereas, for example, the OLS estimate is a point estimate of the population coefficient. The CI can be seen as a possible measure of the precision of the point estimate. That is, once we obtain a point estimate, for example $\hat{\beta}_k$, we ask how precise we expect this estimate to be.

A test and a confidence interval are closely related. We reject the null of the t test that $\beta_k = 0$ above, because it lies outside the confidence interval we calculated here.

```
CI_lower_bound_k = B_hat_k-t_c*B_hat_k_SEE;
CI_upper_bound_k = B_hat_k+t_c*B_hat_k_SEE;
```

3. Simulation exercise on CI

3.1. Set the number of simulations

In this exercise a simulation refers to taking a random sample from the population. Since we want to take samples from the population repeatedly, we will be repeating the simulation multiple times. Define the number of simulations using the code presented at the end of the section.

```
clear;
N_sim = 1000;
```

3.2. Set the sample size

Assume that the model of interest is a simple linear regression model that contains an independent variable but, for simplicity, no constant term. We assume that there are `N_obs` observations available for the dependent and the independent variable of the regression. In our simulation, we will take samples from the population every time we simulate a sample dataset. Setting the number of observations to `N_obs` means that we keep the sample size fixed at `N_obs` each time we draw a sample from the population.

```
N_obs = 1000;
```

3.3. Generate data for the independent variable

The code presented at the end of this section draws `N_obs` theoretical observations from the uniform distribution to create an artificial dataset for the independent variable x_1. In our simulation below, we will generate new data for y repeatedly. Here we generate data for x_1 and we will keep it fixed in the simulation exercise. That is, as we will be taking repeated samples from the population, we will be doing this only for y and not for x_1. This means that we keep the random data of x_1 fixed in repeated sampling. Keeping x_1 fixed in repeated sampling is

indeed an assumption we make. However, note that this is the classical assumption we make while we derive the basic econometric theory. That is, we condition on the values of a regressor while we make econometric derivations. We do this because it simplifies the derivations, and the basics of econometric theory does not change.

```
x_1 = unifrnd(-1,1,N_obs,1);
X = x_1;
```

3.4. Set the true value of the population coefficient of interest

Assume that the population coefficient of the only independent variable β_k is equal to 0.5. This is an assumption of the simulation exercise. In reality we do not observe β_k.

Furthermore, define the number of coefficients to be estimated and assign it to the scalar array `N_par`. We will use `N_par` as an input argument in other functions that produce output.

```
B_true_k = 0.5; % k = x_1.
N_par = 1;
```

3.5. Create an empty matrix for storing coefficient estimates from repeated sampling

Create an empty column vector that will store the estimates of the population coefficient in repeated simulations. Name this column vector as `B_hat_k_sim`. The dimension of the vector is `N_sim` $\times$ `N_par`. That is, the column vector has `N_sim` rows because we will estimate the population coefficient `N_sim` times. We have one population coefficient to estimate, and therefore we consider an empty vector instead of an empty matrix, which is why `N_par` has been set to `1`.

Furthermore, create an empty column vector that will store the simulated standard error estimates of the `N_sim` coefficient estimates.

```
B_hat_k_sim = NaN(N_sim,N_par);
B_hat_k_SEE_sim = NaN(N_sim,N_par);
```

3.6. Create a sampling distribution for the OLS coefficient estimate

In this section we draw `N_sim` random samples from the population as if we could do this in reality. Each sample leads to a coefficient estimate for `x_1`. This leads to a distribution for the OLS estimate of `x_1`.

Presented at the end of the section is a for loop that takes random samples from the population, and estimates the population coefficient of interest using each sample. The first line of the for loop is the index of the for loop that instructs the for loop to execute a program (still to be specified) `N_sim` times.

In the second line of the for loop, we draw random numbers for the error term assuming that the error term follows a standard normal distribution. We specify the dimension of the dependent variable as `N_obs by 1`.

In the third line of the for loop, we generate data for the dependent variable using the assumed true value for the population coefficient and the generated data for the error term. At each iteration of the for loop a new dataset is created for the dependent variable.

In the fourth line of the for loop, we estimate the regression equation using the data generated for `y` and `X`.

In the fifth line of the for loop, we instruct MATLAB to store the coefficient estimate from iteration i in the row number i of the vector array `B_hat_sim`. The sixth line of the for loop repeats the same operation for the standard error estimate of the coefficient estimate. The last line closes the for loop.

```
for i = 1:N_sim
    u = normrnd(0,1,N_obs,1);
    y = X*B_true_k+u;
    LSS = exercisefunction(y,X);
    B_hat_k_sim(i,1) = LSS.B_hat(1,1);
    B_hat_k_SEE_sim(i,1) = LSS.B_hat_SEE(1,1);
end
```

3.7. Calculate the critical t value

Obtain the critical t value using the degrees of freedom, and assuming that we are interested in the 95 percent confidence interval. We will use the critical t value to construct the confidence interval.

```
t_df = N_obs-size(X,2);
t_c = tinv(0.975,t_df);
```

3.8. CI for the population coefficient when we have one sample from the population

In a preceding section we have drawn `N_sim` samples from the population. We can select one of these samples, and construct the confidence interval for the population coefficient of the independent variable. In fact, this corresponds to a real life situation where we have access to a single sample data drawn from the underling population. We can interpret the constructed confidence as follows. In repeated sampling, the probability that intervals like the one constructed based on the sample data will contain the true β_k is 95 percent.

```
CI_lower_bound_k_one_sample = B_hat_k_sim(1,1)-t_c*B_hat_k_SEE_sim(1,1);
CI_upper_bound_k_one_sample = B_hat_k_sim(1,1)+t_c*B_hat_k_SEE_sim(1,1);
```

3.9. Calculate the fraction of the time the population coefficient falls into the random intervals constructed using repeated samples

In the preceding section, the confidence interval has been interpreted as 'in repeated sampling, the probability that intervals like the one constructed based on the sample data will

contain the true β_k is 95 percent'. In our simulation above, we in fact did repeated sampling. This means that we can now indeed construct the random intervals based on the repeated samples. This is done in the first two lines of the code presented at the end of the section.

Note that the end points of the intervals are random aa discussed above. We could draw different samples from the population, and the intervals would be different. The interval based on the sample data is not random, however. It is given based on the sample data. It is a realised outcome of the theoretical random repeated sampling procedure. That is why we call the intervals based on repeated sampling 'random intervals'.

Now, create a dummy variable that takes a value of 1 if the true population coefficient falls within the constructed intervals. We can do this using the third line of the code presented at the end of the section.

We will now calculate the fraction of the time the population coefficient falls into the intervals constructed using the repeated samples. Before doing, so, state the fraction you expect to obtain. Now carry out the calculation. This calculation can easily be done by calculating the mean of the dummy variable just created. What is the fraction you obtained? Is this the fraction you have expected?

```
RI_lower_bound_k_repeated_sample = B_hat_k_sim-t_c*B_hat_k_SEE_sim;
RI_upper_bound_k_repeated_sample = B_hat_k_sim+t_c*B_hat_k_SEE_sim;
B_true_within_interval_dummy = B_true_k > RI_lower_bound_k_repeated_sample & ...
                               B_true_k < RI_upper_bound_k_repeated_sample;
Fraction_B_true_within_intervals = mean(B_true_within_interval_dummy); % The
fraction is equal to (aprox.)  0.95.  Are you surprised?
```