Exercise – Understanding the confidence interval (CI) using simulation

1. Aim of the exercise

The rationale behind a CI is based on the concept of repeated sampling from the population. However, in practical scenarios, we cannot repeatedly sample from the population, which makes the concept somewhat abstract and challenging to teach and learn. Consequently, students often struggle to interpret a CI. Through a simulation exercise, we can mimic the process of repeatedly sampling from the population, thereby concretely demonstrating what a CI represents.

2. Theory

Assume that the OLS estimator $\hat{\beta}_k$ follows a normal distribution with mean β_k and variance $\sigma^2_{\hat{\beta}_k}$:

$$\hat{\beta}_k \sim N\left[\beta_k, \sigma^2_{\hat{\beta}_k}\right].$$

We can standardize $\hat{\beta}_k$ so that it has a standard normal distribution:

$$\frac{\hat{\beta}_k - \beta_k}{\sigma_{\hat{\beta}_k}} \sim N\left[0, 1\right].$$

Suppose that we are interested in the null hypothesis

$$H_0 : \beta_k = \beta_k^0$$

against the alternative

$$H_1 : \beta_k \neq \beta_k^0.$$

Then, under the null hypothesis, we have

$$\frac{\hat{\beta}_k - \beta_k^0}{\sigma_{\hat{\beta}_k}} \sim N\left[0, 1\right].$$

$\sigma_{\hat{\beta}_k}$ is an unobserved population parameter. Replace it with its unbiased estimator $s_{\hat{\beta}_k}$. This gives

$$\frac{\hat{\beta}_k - \beta_k^0}{s_{\hat{\beta}_k}} \sim t\left[n - K\right]$$

where $n - K$ is the degrees of freedom of the t distribution. Then, we can state that

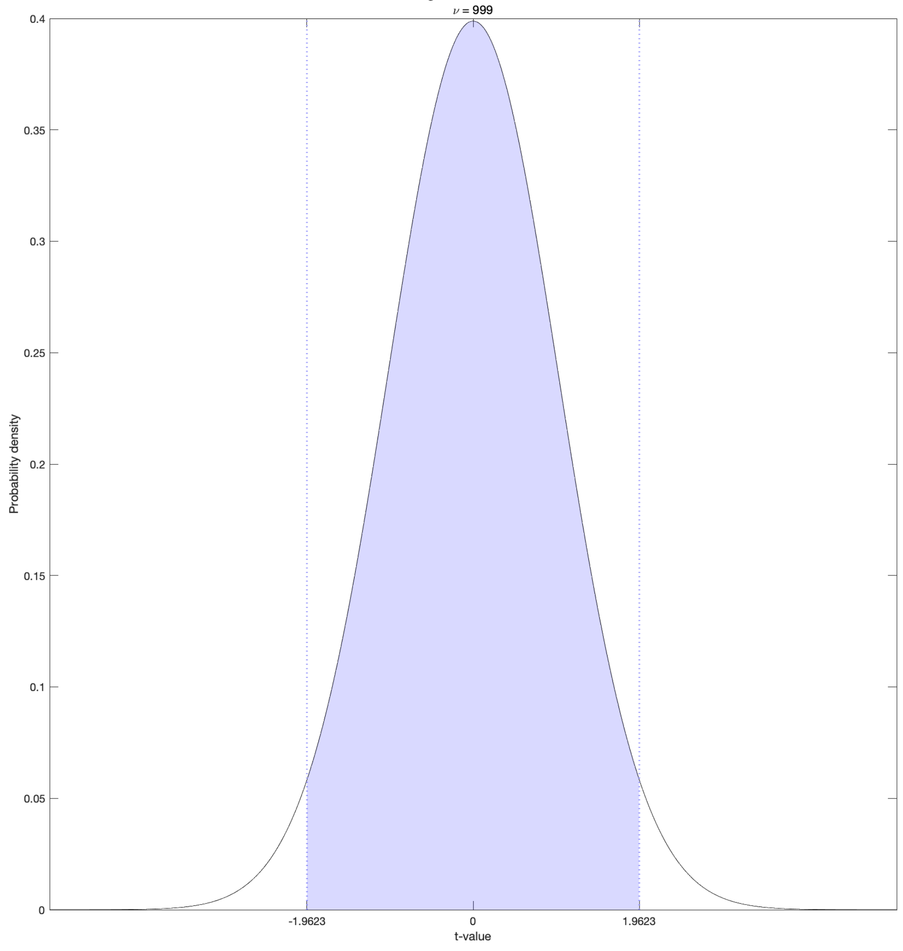
$$\text{Prob}\left(-t_{\alpha/2} < \frac{\hat{\beta}_k - \beta_k^0}{s_{\hat{\beta}_k}} < t_{\alpha/2}\right) = 1 - \alpha$$

α is some probability of our choice. $-t_{\alpha/2}$ and $t_{\alpha/2}$ are some lower and upper thresholds. They correspond to points on the t distribution. The interpretation of this expression is that the probability that the random variable $\frac{\hat{\beta}_k - \beta_k^0}{s_{\hat{\beta}_k}}$ is between the stated thresholds is 95%. For example, if $n - K$ is 999 and α is 0.05,

$$\text{Prob}\left(-1.9623 < \frac{\hat{\beta}_k - \beta_k^0}{s_{\hat{\beta}_k}} < 1.9623\right) = 0.95.$$

Figure 1 illustrates this. It plots the PDF of the t distribution with 999 degrees of freedom. The shaded area between the stated thresholds is 95%.

Fig. 1. t-distribution



Consider again the probability expression

$$\text{Prob}\left(-t_{\alpha/2} < \frac{\hat{\beta}_k - \beta_k^0}{s_{\hat{\beta}_k}} < t_{\alpha/2}\right) = 1 - \alpha.$$

Rearranging the terms, the stated probability is equivalent to

$$\text{Prob}\left(\hat{\beta}_k - t_{\alpha/2} s_{\hat{\beta}_k} < \beta_k^0 < \hat{\beta}_k + t_{\alpha/2} s_{\hat{\beta}_k}\right) = 0.95.$$

At this instance the interpretation changes. The interpretation is for the unique nonrandom population parameter β_k^0. The end points of the interval

$$\left[\hat{\beta}_k - t_{\alpha/2} s_{\hat{\beta}_k}, \hat{\beta}_k + t_{\alpha/2} s_{\hat{\beta}_k}\right]$$

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. 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^0 is in this interval is either 0 or 1. Hence, it is incorrect to say that the probability that β_k^0 is in the interval we estimate is 95%. The interval we computed (using one sample at hand) is just an estimate of one of those intervals (that result from repeated sampling) that contain β_k^0 95 percent of the times. The correct interpretation is the following: In repeated sampling, the probability that intervals like the one we estimate will contain the true β_k^0 is 95%. The probability that this particular non-random interval includes the true β_k^0 is either 0 or 1.

The CI 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 CI are closely related. We reject a null of the t test that $\beta_k^0 = 0$ because it lies outside the CI we calculate that does not include 0.

3. Coefficient and standard error estimates (SEE) from repeated samples

3.1. Clear the memory

Clear the memory from possible calculations from an earlier session.

```
% 3.1. Clear the memory
clear;
```

3.2. Set the sample size

Assume that the model of interest is a simple linear regression model that contains an independent variable and, for simplicity, no constant term. We assume that there are `N_obs` observations available for this independent variable and for the dependent variable of the regression. In our simulation exercise below, we will draw samples from the population. 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.

```
% 3.2. Set the sample size
N_obs = 1000;
```

3.3. Generate data for the independent variable

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

Keeping X fixed in repeated sampling is indeed an assumption we make. Note, however, 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.

```matlab
% 3.3. Generate data for the only independent variable
X = random('Uniform',-1,1,[N_obs 1]);
```

3.4. Define the number of coefficients to be estimated

Define the number of coefficients to be estimated and assign it to the scalar array `N_par`. We will use `N_par` in few occasions in our code below.

```matlab
% 3.4. Define the number of coefficients to be estimated
N_par = size(X,2);
```

3.5. Set a (hypothetical) value for the population coefficient

Assume that the population coefficient of the only independent variable β – `B_true` in the code – is equal to 0.5. We need this to generate values for y. In principle we do not observe β.

```matlab
% 3.5. Set a (hypothesized) value for the population coefficient
B_true = 0.5;
```

3.6. 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. Here we define the number of simulations or samples.

```matlab
% 3.6. Set the number of simulations
N_sim = 300;
```

3.7. Preallocate a matrix for storing OLS estimates from all samples

Create an empty matrix that will store the OLS coefficient estimates of β. Since we will draw `N_sim` samples from the population, we will obtain `N_sim` coefficient estimates based on these samples. Since we have only one coefficient to estimate, that is since `N_par` is 1, the matrix in this case is in fact a column vector.

```matlab
% 3.7. Preallocate a matrix for storing OLS estimates from all samples
B_hat_sim = NaN(N_sim,N_par);
```

3.8. Preallocate a matrix for storing SSEs from repeated samples

Create an empty column vector that will store the `N_sim` standard error estimates of the OLS coefficient estimates of β from repeated samples.

```
32 % 3.8. Preallocate a matrix for storing SSEs from repeated samples
33 B_hat_SEE_sim = NaN(N_sim,N_par);
```

3.9. Coefficient and standard error estimates (SEE) from repeated samples

Here we draw `N_sim` random samples from the population as if we could do this in reality. Each sample leads to an estimate of β. This leads to a distribution of the OLS estimate, referred to as the sampling distribution of the OLS estimate of β – see the exercise on the sampling distribution of the OLS estimate.

We use a for loop to mimic drawing random samples from the population, and estimate the population coefficient of interest using each sample. In the code below, line 36 is the index of the for loop that instructs the for loop to execute the program we are to specify below `N_sim` times. In line 38, we draw random numbers for the error of the regression assuming that the errors follow a standard normal distribution. We specify the dimension of the errors as `N_obs by 1`. In line 40, we generate data for the dependent variable using the assumed true value for the population coefficient, the generated data for X, 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 line 42, we estimate the regression equation using the data generated for `y` and `X`. For this purpose, we utilize an external function that takes a dependent variable and a matrix of independent variables as input arguments, and returns standard OLS statistics as output. In line 44, we store the coefficient estimate from iteration `i` of the for loop in row number `i` of the vector array `B_hat_sim`. Line 46 does this for the standard error estimate of the coefficient estimate. The last line closes the for loop.

Note that if we had samples of y and X at our disposal from repeated samples from the population, we would not need any of the code up to this point and instead we could start the exercise directly with the next section.

```
35 % 3.9. Coefficient and standard error estimates (SEE)
36 for i = 1:N_sim
37     % Draw new error for each sample (in each iteration of the loop)
38     u = random('Normal',0,1,[N_obs 1]);
39     % Generate values for the dependent variable
40     y = X*B_true+u; % The data generating process (DGP)
41     % Obtain OLS statistics using the external function
42     LSS = exercisefunctionlss(y,X);
43     % Store the OLS estimates
44     B_hat_sim(i,1) = LSS.B_hat(1,1); % B_hat is a random variable
45     % Store the SSEs of OLS estimates
46     B_hat_SEE_sim(i,1) = LSS.B_hat_SEE(1,1);
47 end
```

4. Construct random intervals (RIs) from repeated samples from the population

We want to construct RIs. This requires to specify a significance level. Since we want to construct a 95% CI, we take it as 5%. Calculate the degrees of freedom given the number of observations and parameters to estimate. Using these, calculate the critical value from the t distribution. Next, construct the confidence intervals. Note that the dimension of `RIs` is

N_sim by 2. That is, there are N_sim intervals resulting from N_sim samples. 2 is for the lower and upper bounds of the intervals. In a real life scenario, however, we typically have only one sample and hence we can estimate only one CI. In line 65 we extract such a CI from RIs we estimated.

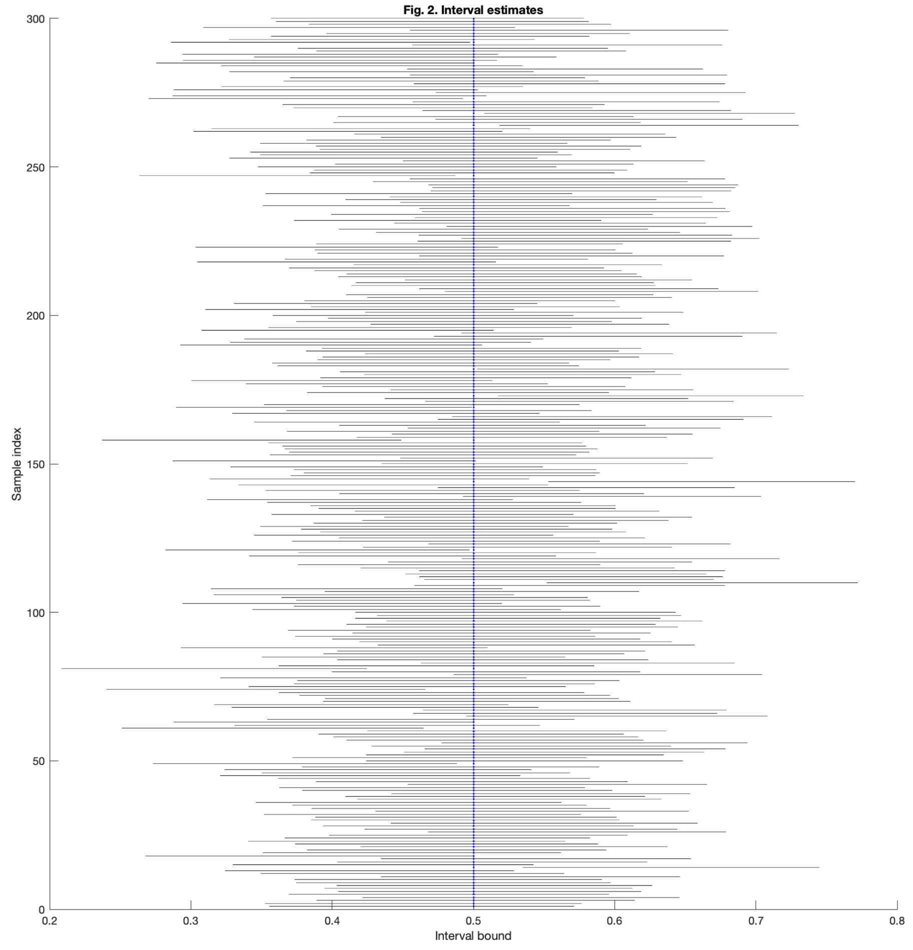
```
49 %% 4. Construct RIs from repeated samples from the population
50 
51 % 4.1. Define the significance level
52 alpha = 0.05; % For 95% CI. Change to 0.10 for 90% CI.
53 
54 % 4.2. Calculate the degrees of freedom for the t distribution
55 df = N_obs-N_par;
56 
57 % 4.3. Calculate the critical value from the t distribution
58 t_c = tinv(1-alpha/2,df);
59 
60 % 4.4. Construct the RIs for B_true using its estimates from all
61 % samples
62 RIs = [B_hat_sim-t_c*B_hat_SEE_sim,B_hat_sim+t_c*B_hat_SEE_sim];
63 
64 % 4.5. Construct the CI for B_true when there is one sample available
65 CI = RIs(1,:);
```

5. Plot the RIs from all samples and the population coefficient

Here we plot the RIs we have estimated using the samples we have drawn from the population. We also plot the population coefficient.

```
66 %% 5. Plot the RIs from all samples and the population coefficient
67 
68 % 5.1. Plot the RIs from all samples and the population coefficient
69 hold on
70 for i = 1:N_sim
71     plot(RIs(i,:),[i,i],'k-','LineWidth',0.5);
72     plot(B_true,i,'bo','MarkerSize',1.5,'MarkerFaceColor','b');
73 end
74 title('Fig. 2.Interval estimates');
75 ylabel('Sample index');
76 xlabel('Interval bound');
77 hold off
```

Fig. 2. Interval estimates



6. Interpret the CI

Figure 2 shows all the RIs we estimated using repeated sampling, and the population coefficient. We can calculate the proportion of the times the population coefficient falls into these intervals. In line 82 we create a dummy variable that takes a value of 1 if the population coefficient falls within the intervals. In line 85 we calculate the proportion of the times the population coefficient falls into the intervals constructed using the repeated samples. The proportion we obtain is approximately 95%. It is not exactly 95% due to simulation noise.

We can now interpret the CI. In repeated sampling, the probability that intervals, like the one estimated using only one sample in line 65, contains the population coefficient β is 95%. The particular CI we have is called "confidence" interval because we use this one and only one interval to be confident about the population coefficient with some probability.

```
79 %% 6. Interpret the CI
80 
81 % 6.1. Create a dummy indicating if B_true is within the RIs
82 B_true_is_within_RIs = B_true > RIs(:,1) & B_true < RIs(:,2);
83 
84 % 6.2. Calculate the proportion of times B_true is within the RIs
```

```
Proportion_B_true_is_within_RIs = mean(B_true_is_within_RIs); % App.
% 0.95
```

7. Final notes

This file is prepared and copyrighted by Renata-Maria Istrătescu and Tunga Kantarcı. This file and the accompanying MATLAB files are available on GitHub. They can be accessed via this link.