Exercise – Understanding serial correlation using simulation

1. Aim of the exercise

To understand the implications of violating the spherical errors assumption with serial correlation for the sampling distribution of the OLS estimator using simulation.

2. Theory

A problem that can arise with an OLS model is serial correlation where one or more of a model's errors influence one or more other errors. This is most often considered in the context of time-series analysis where past values of the error term influence current values. The presence of serial correlation by itself is an efficiency problem, similar to that with heteroskedasticity. MATLAB has the `arima` function for time-series analysis which we can use to simulate a time-series process. In particular, we will use the function to create an AR(1) process, which is serial correlation in which the value of the error term in the last period influences the current value, but values beyond one period back do not have any additional unique effect on the current value.

3. Application

3.1. Clear the memory

Clear the memroy.

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

3.2. Set the number of simulations

Set the number of simulations to be carried out.

```
% 3.2. Set the number of simulations
N_sim = 1000;
```

3.3. Set the sample size

Assume a linear regression model that contains a column of 1s and an independent variable. In this simulation exercise we will consider a time-series specification for the errors of the model. We set the sample size to 50, which could represent, for example, 50 years of annual data.

```
% 3.3. Set the sample size
N_obs = 50;
```

3.4. Define the number of parameters to be simulated

Define the number of parameters to be simulated.

```
% 3.4. Define the number of parameters to be simulated
N_par = 3;
```

3.5. Set true values for the coefficients of the intercept and the independent variable

We assume that we know the true values of the coefficients of the variables of the linear regression model.

```
% 3.5. Set true values for the coefficients of the model
B_true = [0.2 0.5]';
```

3.6. Generate data for the systematic component of the regression model

Generate data for the systematic component of the regression model.

```
% 3.6. Generate data for the systematic component of the model
x_0 = ones(N_obs,1);
x_1 = random('Uniform',-1,1,[N_obs 1]);
```

3.7. Preallocate empty matrices for storing simulated coefficients

Preallocate an empty matrix to store in each of its columns `N_sim` simulated coefficient estimates from `N_par` regression models. Therefore, the matrix is `N_sim` $\times$ `N_par`.

```
% 3.7. Preallocate empty matrices for storing simulated coefficients
B_hat_1_sim = NaN(N_sim,N_par);
```

3.8. Create the sampling distribution of the OLS estimator

Here we consider a for loop to create sampling distributions for the OLS estimator under three model specifications. In the first model we assume that the errors of the model are normal. In the second model we assume that they are AR(1) where we assume an auto-correlation parameter value of 0.75. This could vary between 0 and 1, with larger values indicating stronger serial correlation. There are many possible solutions and strategies for dealing with serial correlation, the choice of which should depend on what we think the underlying DGP is. In the third model, we focus on the popular modeling strategy of including a lagged value of the dependent variable in the model as an independent variable.

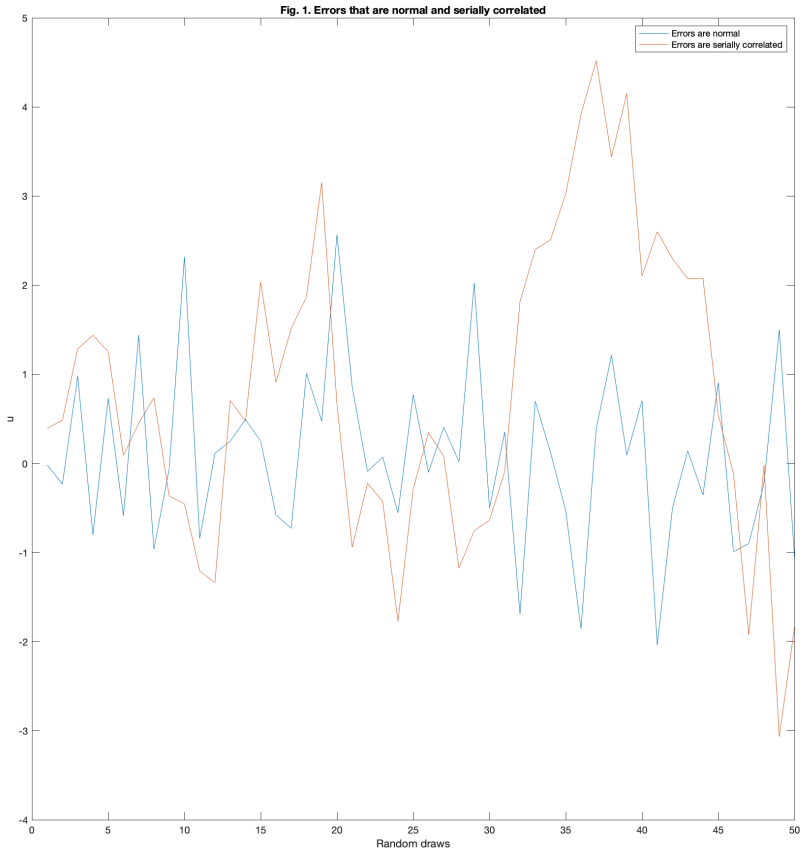
```
% 3.8. Create the sampling distribution of the OLS estimator
for i = 1:N_sim
    model = arima('Constant',0,'AR',0.00,'MA',0,'Distribution', ...
        'Gaussian','Variance',1);
    u = simulate(model,N_obs);
    X = [x_0 x_1];
    y = X*B_true+u;
    LSS = exercisefunctionlss(y,X);
    B_hat_1_sim(i,1) = LSS.B_hat(2,1);
    model = arima('Constant',0,'AR',0.85,'MA',0,'Distribution', ...
        'Gaussian','Variance',1);
    u_ar = simulate(model,N_obs);
    X = [x_0 x_1];
    y_ar = X*B_true+u_ar;
    LSS = exercisefunctionlss(y_ar,X);
```

```
    B_hat_1_sim(i,2) = LSS.B_hat(2,1);
    y_ar_lag = lagmatrix(y_ar,1);
    X = [x_0 x_1 y_ar_lag];
    LSS = exercisefunctionlss(y_ar(2:N_obs,:),X(2:N_obs,:));
    B_hat_1_sim(i,3) = LSS.B_hat(2,1);
end
```

4. Plot normal and serially correlated errors

Plot serially correlated and normal errors.

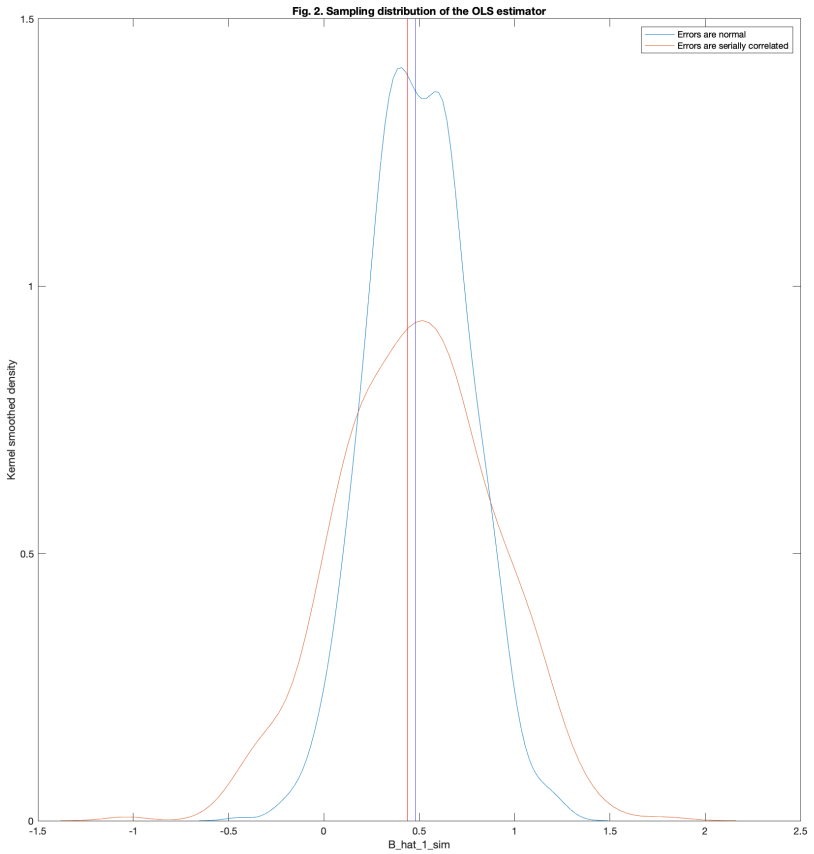
```
%% 4. Plot normal and serially correlated errors
figure
hold on
plot(u)
plot(u_ar)
title('Fig. 1. Errors that are normal and serially correlated')
legend('Errors are normal','Errors are serially correlated')
ylabel('u')
xlabel('Random draws')
hold off
```



5. Plot the sampling distribution of the OLS estimator when errors are serially correlated and when they are not

Plot the sampling distributions of the OLS estimator when errors are serially correlated and when they are not. What do you conclude?

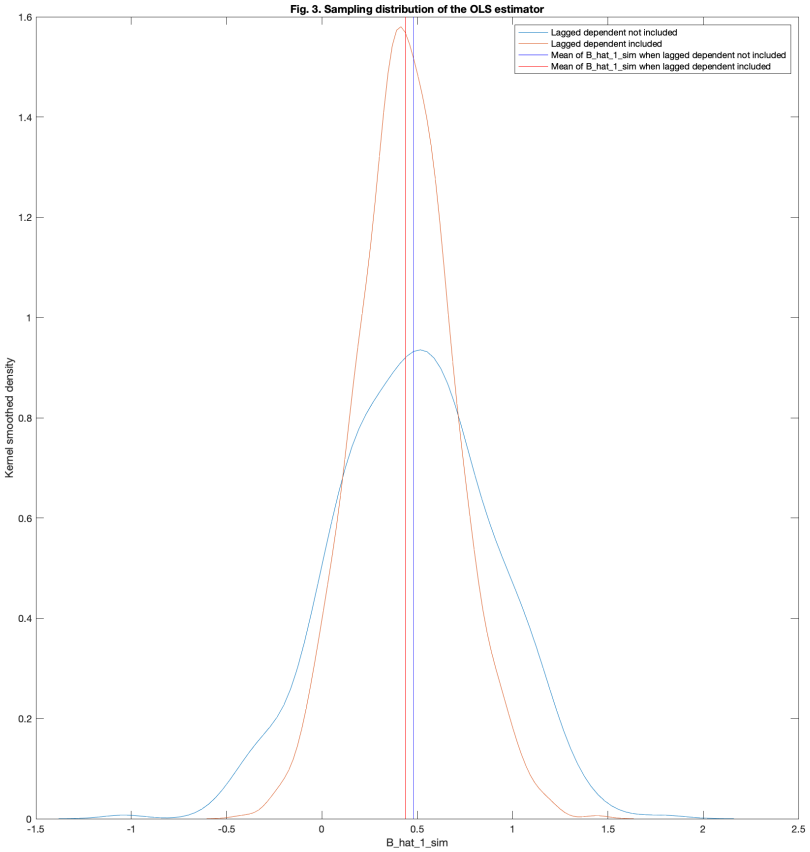
```
%% 5. Sampling distribution of the OLS estimator and serial correlation
figure
hold on
ksdensity(B_hat_1_sim(:,1))
ksdensity(B_hat_1_sim(:,2))
title('Fig. 2. Sampling distribution of the OLS estimator')
legend('Errors are normal','Errors are serially correlated')
ylabel('Kernel smoothed density')
xlabel('B\_hat\_1\_sim')
hold off
```



6. Plot the sampling distribution of the OLS estimator when the regression includes the lagged dependent variable and when it does not

The plot shows the distribution of the OLS estimates when the lagged dependent is included in the regression model and when it is not included. The result is a trade-off between bias and efficiency. The estimator of the model that includes the lagged dependent variable is more efficient: the peak of the distribution of the estimated values is higher, and the distribution is less spread compared to the distribution of the estimates for the model that does not include the lagged dependent variable. However, the peak is slightly off from the true value. In contrast, the distribution of the estimates from the OLS regressions that does not include the lagged dependent variable as an independent variable is centered right at 0.50. The spread of the estimates for this model is larger, however, indicating less efficiency. This shows that including a lagged value of the dependent variable as an independent variable in the model improves efficiency because it captures serial correlation in the residual, but at the cost of some bias. This bias emerges because the lagged value of the dependent has a random component to it. That is, by definition, the dependent variable consists of a systematic and a stochastic component that leads to some level of bias.

```
%% 6. Sampling distribution of the OLS estimator and lagged dependent
figure
hold on
ksdensity(B_hat_1_sim(:,2))
ksdensity(B_hat_1_sim(:,3))
line([mean(B_hat_1_sim(:,2)) mean(B_hat_1_sim(:,2))],ylim, ...
    'Color','blue')
line([mean(B_hat_1_sim(:,3)) mean(B_hat_1_sim(:,3))],ylim, ...
    'Color','red')
title('Fig. 3. Sampling distribution of the OLS estimator')
legend('Lagged dependent not included','Lagged dependent included', ...
    'Mean of B\_hat\_1\_sim when lagged dependent not included', ...
    'Mean of B\_hat\_1\_sim when lagged dependent included')
ylabel('Kernel smoothed density')
xlabel('B\_hat\_1\_sim')
hold off
```



7. Final notes

This file is prepared and copyrighted by Tunga Kantarcı. Parts of the simulation exercise are based on Carsey, T. M., and Harden, J. J., 2014. Monte Carlo simulation and resampling methods for social science. SAGE Publications. This file and the accompanying MATLAB files are available on GitHub and can be accessed via this link.