Macroeconometrics Problem Set 1: Explanatory Notes

Corti Pietro (3061891), Reichlin Vito (3081010)

24 October 2022

Point 1

a) To write down an $AR(1)$ process, i.e.

$$y_t = \phi y_{t-1} + \epsilon_t$$

with a for loop we decided to set the number of observations $N = 500$ and then set the initial values for running the command, in particular we have defined a vector with 500 rows and 1 column respecting the requisites of the error term (eps1) and the initial value for y_t, given by Yt1. The command that you can see on the matlab file gave us the result.

b) To use this command we exploit the fact that an $AR(1)$ process can be written as an $MA(\infty)$; this condition is possible since we know that $|\phi| < 1$. This process is indeed necessary to respect the inputs of the command "filter" that allows normally to write an MA process; however, knowing that this $AR(1)$ can also be written in this way:

$$y_t = (1 - \phi L)^{-1}\epsilon_t = \sum_{j=0}^{\infty} \phi^i L^i \epsilon_t$$

we can set at the denominator input the value $(1 - 0.4)$, at the numerator the input 1 and eps1 as the generating process. To obtain the process we would like to study.

c) Here we just show the graph of the two AR processes; as we can see, putting the same initial condition (given by the same command eps1 for the two processes) we are able to see that they are perfectly overlapping.

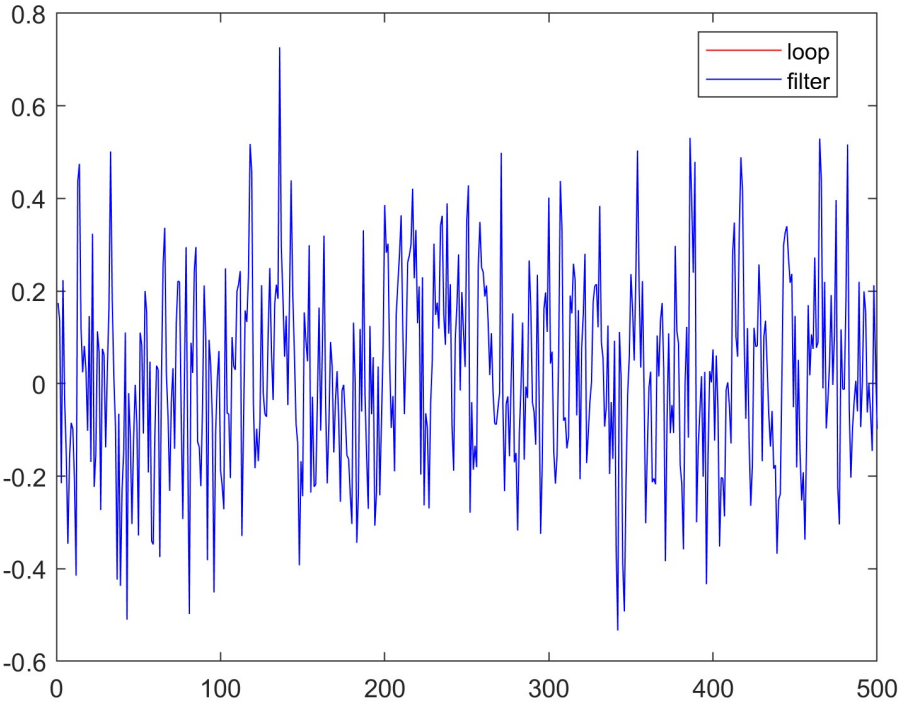

Figure 1: AR(1) processes with the loop and "filter". Notice that the red line is perfectly overlapped by the blue line, this is because the two processes are equivalent.

Point 2

We start with an AR(1) process (see Point 1 for further explanations). We set the initial condition $y_0 = 10$. The next observation will be: $y_1 = \phi 10 + \epsilon_t$. Now, if we iterate the process we end up with:

$$y_T = 10\phi^T + \sum_{j=0}^{T-1} \phi^i \epsilon_{T-1}$$

If we take the unconditional mean of this process we have:

$$\mathbb{E}(y_t) = 10\phi^T + \mathbb{E}\left\{ \sum_{j=0}^{T-1} \phi^i \epsilon_{T-1} \right\} = 10\phi^T + \mathbb{E}(\epsilon) \sum_{j=0}^{T-1} \phi^i$$

as $T \to +\infty$, $10\phi^T \to 0$ since $\phi = 0.4 < 1$. So we end up only with the last term, and applying the rule of the geometric series we get:

$$\mathbb{E}(y_t) = \frac{\mathbb{E}(\epsilon)}{1-\phi} = \frac{\mathbb{E}(\epsilon)}{1-0.4} = 3$$

In order to have an unconditional mean of the process equal to 3, we solve for:

$$\mathbb{E}(\epsilon) = 3(1-0.4) \longrightarrow \mathbb{E}(\epsilon) = 1.8$$

Therefore, we let the white noise to be distributed as a normal with variance 0.2 and mean 1.8. This will yield the desired unconditional mean.

Then we are asked to insert the initial condition equal to 10; if we do this inside the loop for the AR(1) (see the Matlab file for complete code) we get the figure below (Figure 2). Basically we substitute the first value of "eps" with 10, and then started the loop. As we can see from the picture when the initial condition is far from the unconditional mean process, we get a shift in the process; this means that we are no more around the mean of the process itself (in this case 3). To adjust the problem we should truncate the process after the first observation, which is like an outlier, and consider only the parts in which it is stable around the mean. Doing this gives us the next figure of the block (Figure 3).

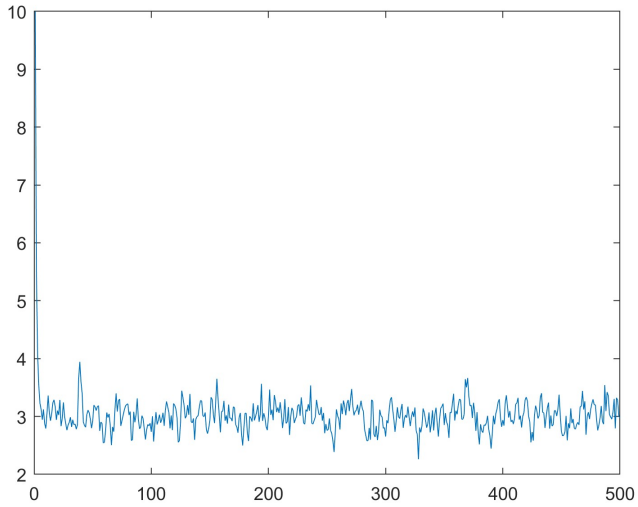

Figure 2: AR(1) processes with the loop and initial condition.

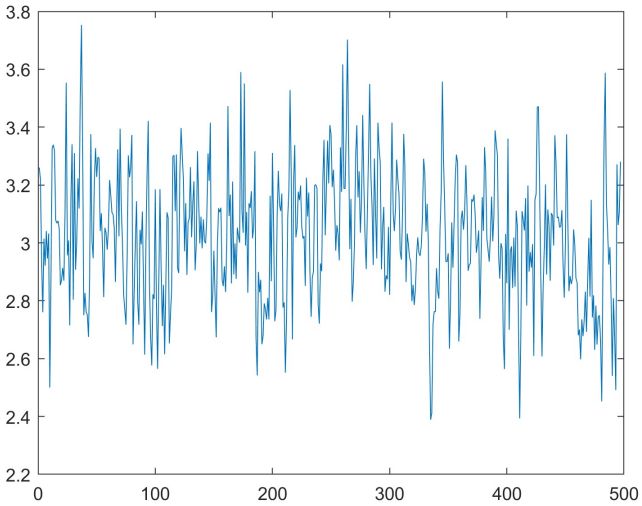

Figure 3: AR(1) processes with the loop corrected

We get exactly the same result if we use the "filter" command (see the matlab file, where we always substituted the first value of the DGP "eps").

Point 3

Recall a MA(1) process:

$$y_t = \theta\epsilon_{t-1} + \epsilon_t$$

a) Using a for loop: We generate a cycle running n number of times where n is the required amount of observations. At each step a random variable is taken from a normal distribution with mean zero and variance σ^2 (in our case $\sigma^2 = 0.3$), these is going to be the current period white noise. Then y_t is computed by summing the newly generated white noise (ϵ_t) and last period white noise (ϵ_{t-1}) multiplied by the MA polynomial coefficient ($\theta = 0.3$). Notice that ϵ_t is then saved in the variable u (see code) so that in the next cycle the lagged white noise will be available in u.

b) Using the filter function: We first generate n white noises taken from a normal distribution with mean zero and variance σ^2. Then, the MA(1) process can be generated from the equation below:

$$y_t = \frac{(1 + \theta L)}{1}\epsilon_t$$

The first input of the filter function is the array of the coefficients for the polynomial in the nominator while the second input corresponds to the denominator. Therefore, we set the first input as 1 and 0.3 while the second just as one. Finally the third term corresponds to the vector of ϵ_t in the equation above, we input the generated white noises.

Point 4

The ARMA function returns the realizations of the ARMA(p,q) process and of the white-noise. The function takes as inputs: the variance of the white noise (var), the number of observations to generate (T), the AR and MA polynomial coefficients (AR and MA) or the AR and MA roots of the corresponding polynomials (ARroots and MAroots). To make sure that the user can input both the polynomial coefficient or its roots we create an initial condition. The condition checks if the inputted coefficients are equal to zero, in this case it assumes that the roots have been inputted instead, and computes the coefficients from the inputted roots. Then it checks if also the roots are equal to zero, in which case it assumes that the user wants to exclude either the MA or AR process and reconverts the value of the coefficients to zero. At the end of the initial conditions the function has clear inputs for both the MA and AR polynomial coefficients.
We then create two arrays, one for the MA and one for the AR, of length T with the first positions occupied by the polynomial coefficients and the rest of the positions occupied by zeros. We also generate two arrays (x and w) of length T only filled with zeros, which will be used to contain the lagged outputs for the AR and the lagged white noises for the MA. With the same purpose we also create two empty arrays (z and u).
We then create a for loop with the following commands. At the beginning of the current period a white noise (e) is generated from a normal with mean zero and the variance that has been inputted. The realization of the current period is then computed, a simple visualization is as follows:

$$y_t = \begin{bmatrix} y_{t-1} y_{t-2} \dots y_1 0 \dots 0 \end{bmatrix}_{1xT} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_p \\ 0 \\ \vdots \\ 0 \end{bmatrix}_{Tx1} + \begin{bmatrix} \epsilon_{t-1} \epsilon_{t-2} \dots \epsilon_1 0 \dots 0 \end{bmatrix}_{1xT} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_q \\ 0 \\ \vdots \\ 0 \end{bmatrix}_{Tx1} + \epsilon_t$$

y_t is then saved as the first value in the array z which is in turn saved (as first elements) in the array x, which will then include all the saved values of the realizations plus a number of zeros such that the x vector has length T and can be premultiplied to the AR coefficients vector. The same procedure happens for ϵ_t which is saved in u and then in w. Notice that the cycle ends when all the positions of the x vector and w vector are filled. These will be respectively the vector containing the realizations and the white noise.

Point 5

a) To complete this point we decided to use a simulation (like a Monte Carlo one) to get the empirical distribution of the OLS estimator for the $AR(1)$. We proceed by identifying the main building-blocks for the multivariate OLS estimator.
Firstly, we set up the number of observations required and we correct the sample size to compute all the statistics (T_eff); then we define the Data Generating Process (DGP) by using a "randn" variable and the corresponding "filter" command to get an AR(1) process (Y_DGP).
To get the lagged observations we install the econometrics toolkit and use the command "lagmatrix" on "Y_DGP" to get the variable "Xt"; we then decide to add the constant term in the regression by adding the "ones" in "Xt", and we correct the process eliminating the first $p + 1$ (in this case 2 since $p = 1$) observations in the DGP. After having adjusted "Y_DGP" for the comformability getting its new variable "Yt", we are ready to compute the OLS estimator (phi_hat) with the corresponding matlab command; as said above, to obtain enough results to compute the empirical OLS distribution we create a loop with 10000 iteration and then create the histogram for phi_hat. The result is the following:

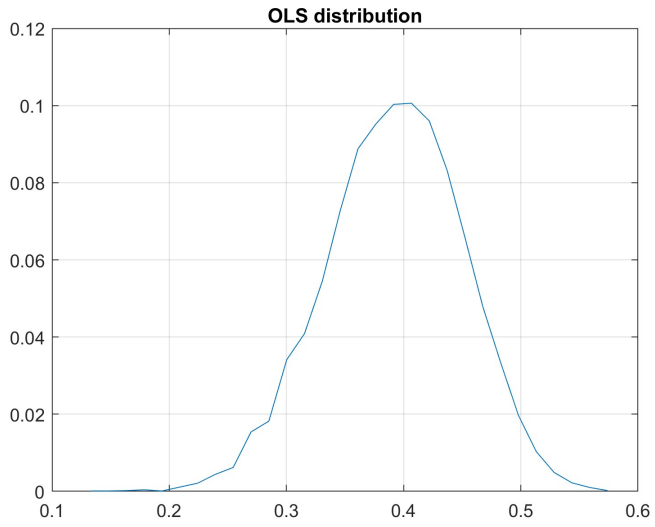

Figure 4: Distribution for the OLS estimator of the AR(1) process

as we expected, the distribution is centered around the true value of ϕ which is equal to 0.4, and it resembles a Normal.

b) To construct a two-tailed t-test for ϕ we have defined in our file all the

necessary elements (see also point a) for further explanations). To explain our results, we present a simplified version of the regression we have computed using this formulation (we don't consider here rigorously temporal indexes just for the simplicity of the exposition):

$$y = X\phi + \epsilon$$

Since we have already compute the OLS, i.e. $\hat{\phi} = (X'X)^{-1}X'y$ we can preoceed by defining the error term (eps_hat) in the following way:

$$\hat{\epsilon}_t = y_t - X_t\hat{\phi}$$

we collect it into a matrix and use it to get the estimator of the variance for the error term ("ee" is the product of the errors and "var_eps_hat" is the variance):

$$\hat{\sigma^2} = \frac{\hat{\epsilon}'\hat{\epsilon}}{T-2}$$

here at the denominator we needed to subtract another time because of the presence of the constant term.
Now, we have the elements we need to compute the variance of the estimator, given that we have also defined the inverse of the product of the X's before ("invXt" in our matab file). To get it we need to compute:

$$Var(\hat{\phi}) = \hat{\sigma^2}\left(\frac{X'X}{T}\right)^{-1}$$

and the std. error is just its square root. It's important to say that to compute this variance in our matlab file (var_phi_hat) we focused on the element of the diagonal of invXt, since we are not interested in the covariances (in the file we used the (2,2) element, but we could also consider the (1,1) element, since they're the same).
Everything is settled down to compute the t-statistics for the two tailed test written in the text:

$$t - stat = \frac{\hat{\phi}_i - 0}{\sqrt{Var(\hat{\phi}_i)}}$$

Lastly, to understand how many times we reject the nulla at the 95% confidence interval we compute the critical values ("critical" in the file) for the inverse of the t-student and and see how many times we have t-statistics bigger than this values (for simplicity we consider 1.96, but the value of "critical" is basically the same); it turns out that we reject the null all the times (see the matlab file to a further exposition of the method we used).

Point 6

To answer this question we simply proceed in the same way of the previous point, but creating a loop for each value of T required by the text (50, 100, 200, 1000). So we have two nested loops, one for the values of T and the other for the simulation (10000 times also here).
The figure below shows our results:

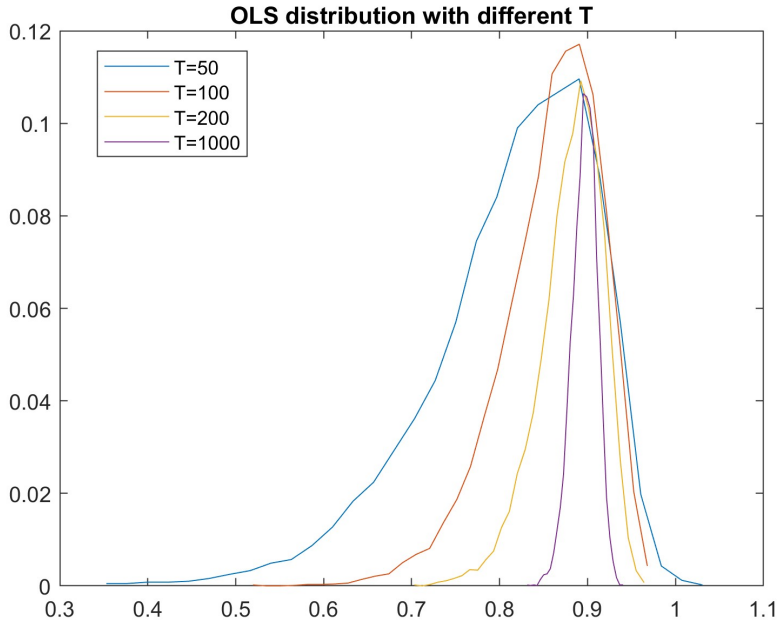

Figure 5: OLS Distribution with different values of T

as you can see, the more time observations we have, the better is the estimation of the OLS in our AR(1) process with $\phi = 0.9$; indeed the three empirical curves gets more concentrated around the value of 0.9, and less dispersed when the size increases.

Point 7

First we set the sample size on which each OLS coefficient is to be estimated on and create an an empty array A where all the OLS are going to be saved. We then begin a for loop with 1000 steps, at each step a new OLS is estimated and saved in A, meaning that A will have length 1000. Inside the for loop we proceed as follows. We generate the vector of white noise from a Normal distribution with mean zero and variance one. We then generate the observations from a MA(1) using the filter function as in Point 3(b). We then create the vectors necessary to run the specified regression:

$$y_t = ay_{t-1} + \epsilon_t$$

The vector of dependent variables in selected from the generated MA(1) observations, excluding the first point in order to allow Y_1 to be regressed on Y_0, where Y_0 is the first generated point of the MA(1). The vector of independent variables instead starts from Y_0 and excludes the last generated point in order to have the same length of the dependent vector. The regression above is then run and the OLS estimate for a is saved in the array A. At the end of the cycle the array A will have 1000 values of a. We find that the mean of the distribution of a's is equal to 0.4404. The histogram is shown below.

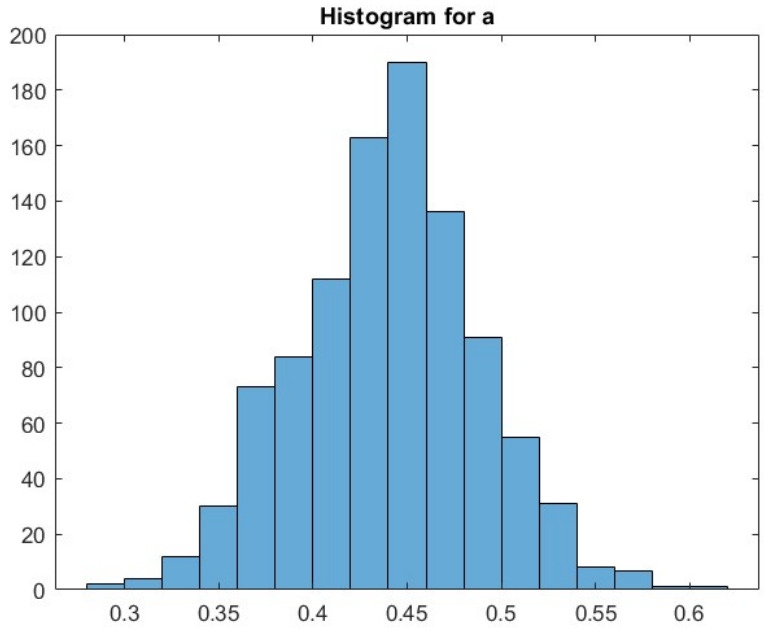

Figure 6: Histogram of the estimated OLS coefficients for $T = 250$

In order to see whether the mean of the OLS estimates converges to some

value for larger and larger sample sizes, the same process as above is repeated in a for loop for increasing values of T. In particular we let T go from 251 to 2001 (increasing by steps of 10) and at each step the mean is saved in the array "means". The plot below shows the means evaluated at each size of the sample (x-axis). For larger sample sizes the mean converges to a value around 0.4412 (last mean computed with the largest sample size), which is close to the initial value found with a smaller sample size.

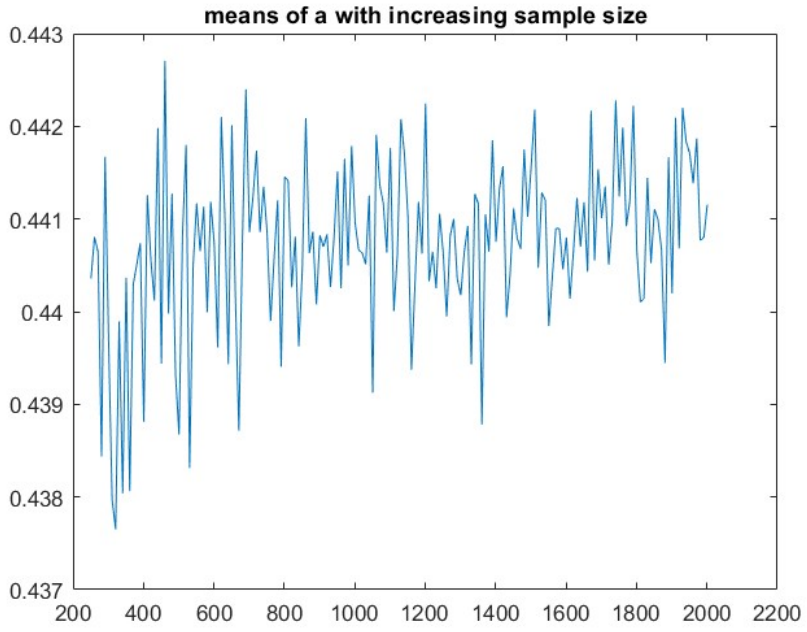

Figure 7: Mean values (y-axis) for the estimated OLS with different sample sizes (x-axis)

Point 8

a) To compute the OLS distribution in the case $\phi = 1$ we proceed in the same way of the previous points, with the same commands. The distribution we get is the following:

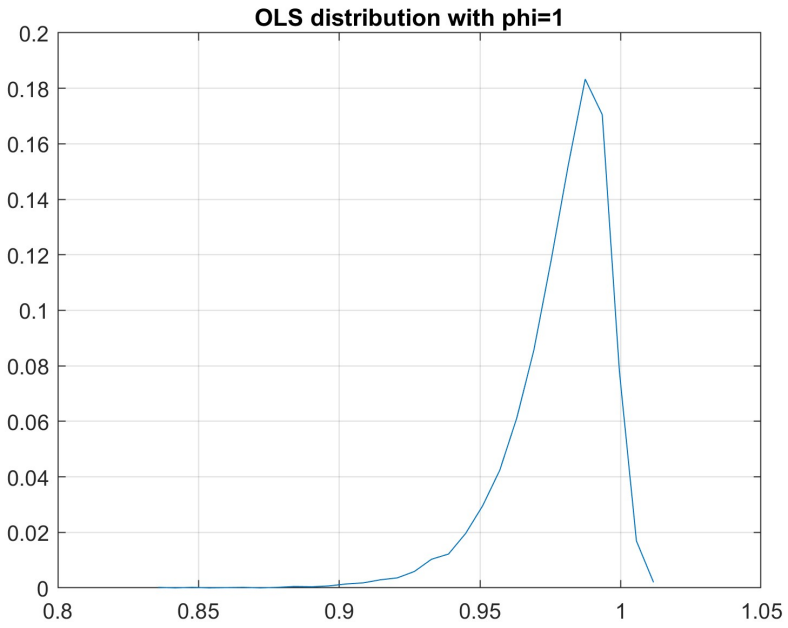

Figure 8: OLS distribution for $\phi = 1$

as we expect, the distribution is not perfectly normal, since now we are in the case of a unit root, and the estimator should follow a DF distribution.

b) As required by the text we now focus on the following regression:

$$\Delta y_t = \alpha + \rho y_{t-1} + \epsilon_t$$

in particular we want to compute a unit root test, i.e. $H_0 : \rho = \phi - 1, \quad \alpha = 0$ and $H_1 : \rho < 1$.
To run this part of the code in our matlab file we have basically used the same commands of before, with the important difference of having subtracted the lag components in the LHS of the regression; in particular we identified a new variable "Y" which gives exactly this result.

The t-stat for this test can be defined as follows:

$$\frac{\hat{\phi} - 1}{\sqrt{Var(\hat{\phi})}}$$

We then proceeded with the computation of the t-stat in a similar fashion of before, using as always 10000 iteration in the loop to show the results.
We ended up with a rejection ratio ("rej_ratio" in the matlab file) of approximately 30%, which is a very different result of the one we can get from the Normal case; but this is because the distributions are actually different. Now the t-stat doesn't follow a Normal distribution, but a DF2 distribution (we distinguish between the asymptotic distribution of $\hat{\phi}$ which is defined as DF1 and the one of this statistic) and we clearly see that it is less skewed than the DF1 and more similar to a Normal (even if not the same). In the next graph we can appreciate the comparison between the DF2 distribution and the one of a N(0,1) (always for 10000 iterations):

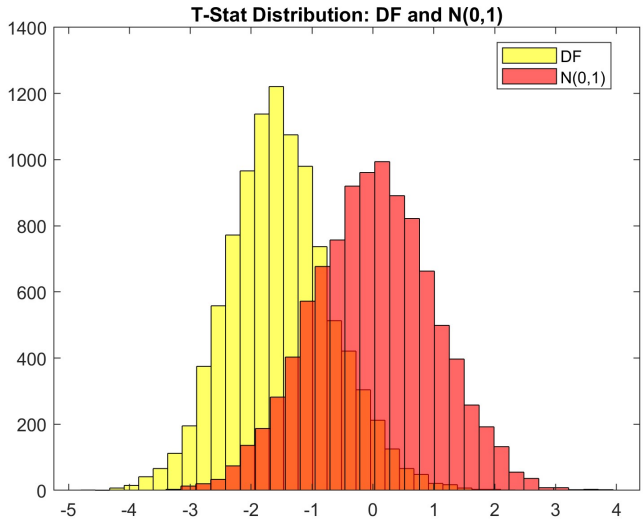

Figure 9: Comparison between DF2 and a N(0,1)

To sum up, we can see that the DF2 distribution is not symmetric as the Normal one, and this is in line with what happens for unit-root testing.

c) To solve this point we just use the command "prctile(t_stat, [1 5 10 25 50 75 90 95])" and we use these indicated critical values (this is the case in wich a constant is considered in the regression). Then, we compare them with the ones in "Applied Econometric Time Series", Wiley Series in Probability and

Mathematical Statistics, John Wiley Sons, 1995, pages 223, 419 and 421, and the ones in James D. Hamilton, "Time Series Analysis", Princeton University Press, 1994, page 763. We used many values, but only the 1%, 5% and 10% are reported as relevant critical values. The following table report our estimates compared to the one computed by Dickey and Fueller.

Percentile	1%	5%	10%
Our Estimates	-3.4980	-2.8695	-2.5732
Enders \Hamilton	-3.51	-2.89	-2.58

As we can see, our values are almost the same of the ones estimated by these authors and this fact straighten the robustness of our results.

Point 9

a) To compute the Monte Carlo (MC) simulation, we proceed basically in the same way of the previous points; indeed a MC simulation is a computational method to obtain numerical results; in our case we use the simulation to obtain the values of the regressions and of the statistical tests computed. We decided to use 5000 iterations ("simu" in our file), just as we did during the TA in class. Now, let's start by a Random Walk (RW) process with a drift (our DGP), defined in this way:

$$y_t = \delta + y_{t-1} + \epsilon_t$$

To understand the empirical distribution of the regression coefficient we consider firstly the following regression:

$$y_t = \delta + \phi y_{t-1} + \epsilon_t$$

Now, to work with it for a DF test as required we rewrite it in this way:

$$\Delta y_t = \delta + \theta y_{t-1} + \epsilon_t$$

where now $\theta = \phi - 1$.

From the previous points we remember that the DF test in this case can be defined as:

$$\frac{\hat{\theta} - (\phi - 1)}{\sqrt{Var(\hat{\theta})}}$$

with $H_0 : \delta = 0, \quad \phi - 1 = 0$.

The commands and variables we used in the matlab file are exactly the same applied for computing empirical distributions in point 8 (we also added constants to the regression as before); the only difference now is that now in the matrix of the regressors ("Xt") we added a third column because of the presence of the drift. The result of this set of commands is given by the following figure:

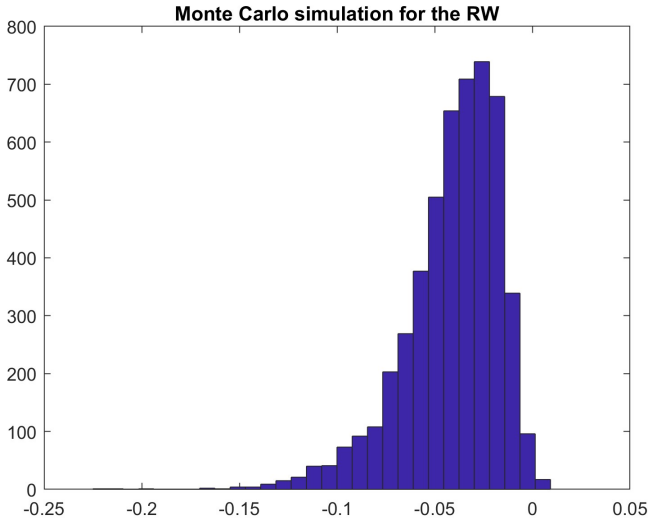

Figure 10: Distribution of the coefficient for the Random Walk

as we can see its distribution resembles the one of a DF test, just like the ones we identified above.

b) To understand how we computed the F-test we firstly need to define it from a theoretical perspective.
Considering again the very simple regression:

$$y = X\beta + \epsilon$$

we can consider a test with $H_0 : \beta = c$ and compute the test for the joint significance of the coefficients with the following statistics:

$$F - stat = \frac{(\hat{\beta} - c)Var(\hat{\beta})^{-1}(\hat{\beta} - c)}{k}$$

and it can be shown that under the null:

$$F - stat \sim F(k, T - k)$$

which is an F-distribution (the ratio of two independent χ^2 distributions) with k and $T - k$ degrees of freedom (dgf).

Applied to our case, the values that this test gets are: $\hat{\beta} = \hat{\theta}$ and $c = \phi - 1 = 0$; the degrees of freedom are: $k = 2$ (since we have an AR(1) with 1 constant in the regression) and $T - k = 248$ because $T = 250$ by the text.
The F-test distribution is the following:

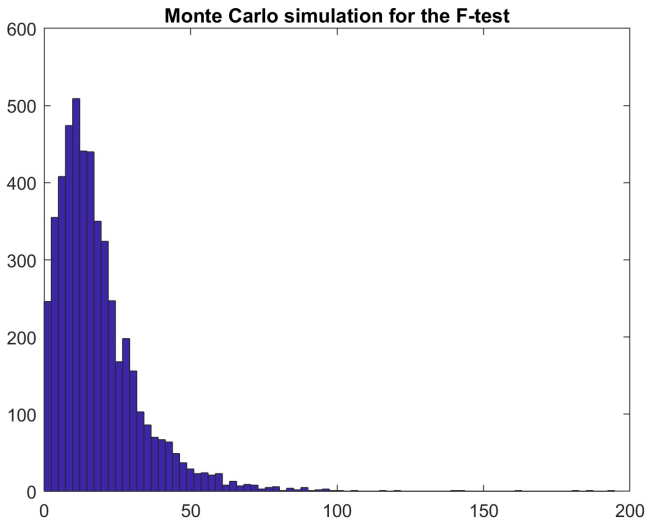

Figure 11: Distribution of the F-test for the RW process

which actually resembles the theoretical distribution for this test.
To compute lastly the number of times we reject the test we needed firstly to find the critical values. We have used the ones you can find in Hamilton (1994) at pag. 764, table B.7 for the DF test; in particular for $T = 250$ and $\alpha = 5\%$ the corresponding value is 6.34. The result, computed on 5000 simulation indicated that we reject the null approximately 83% of the times.

c) A deterministic time trend is defined by the following equation:

$$y_t = \alpha + \delta t + \epsilon_t$$

The commands in this point are the same of the previous MC simulations, with the exception of the DGP; now we used a deterministic time trend, as explained above, to compute it. To be even more precise, within the loop for the MC simulation (indexed by the variable "simu"=5000), we computed the loop for the deterministic time trend creation (we assumed $T = 250$ as in the previous points, and we freely chosen the values of δ and α since not specified by the text).
For the rest, the point is very similar to what we have already done before: we defined all the variables needed for the DF test and the OLS estimation of ϕ as well as the ones needed for the F-test.
To answer the last question we also used its critical values and found that we always reject the null hypothesis. Below we show the distribution of the F-test for the deterministic time trend.

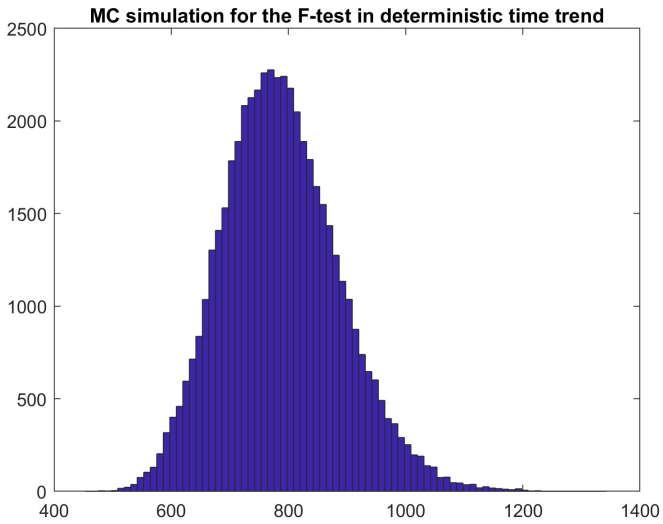

Figure 12: F-test for the deterministic time trend

References

- Bruno, Giovanni (2022), "Econometric Models: Lecture Notes for 2021/2022 ESS"
- Ghysels, Eric Marcellino, Massimiliano (2018), "Applied Economic Forecasting Using Time Series Methods", Oxford University Press
- Enders, Walter (1995), "Applied Econometric Time Series", Wiley Series in Probability and Mathematical Statistics, John Wiley Sons
- Hamilton, James D. (1994), "Time Series Analysis", Princeton University Press
- Sala, Luca (2022), "Notes on Time Series Analysis"