Week02 Project Report

Problem 1

Problem

Remember from last week we discussed that skewness and kurtosis functions in statistical packages are often biased. Is your function biased? Prove or disprove your hypothesis.

Answer

For both the skew and the kurtosis function in `SciPy` Library of Python, we can set the parameter `bias` to either `True` or `False`, to get the "biased" moment or "unbiased" one. However, do the "biased" and "unbiased" functions really get the results as they say?

I tested the two versions of skewness and kurtosis functions using hypothetical test. I found that the two kurtosis functions, no matter setting `bias` to `True` or `False` , are both biased; while the two skewness functions are both likely to be unbiased.

To do T-test, I sampled standard random normal values, 100 each group, to calculate their skewness and kurtosis using the functions in `SciPy.stats` , setting `bias` to `True` and `False` , and repeating this 1000 times to generated arrays for them. I got four 1000-element arrays and did one-sample two-side T-test for them. The null hypothesis is: skewness/kurtosis (setting `bias` = `True` / `False`) is unbiased, which means that $\mu_0 = 0$ for the four tests. (Note that I used standard normal values, so the unbiased skewness and excess kurtosis are expected to be 0.) Here are the results.

T-stat	`bias` **=** `True`	`bias` **=** `False`
skewness	0.28236070877406555	0.2823607087740656
kurtosis	-2.4119631530570347	-2.4119631530570347

P-value	`bias` **=** `True`	`bias` **=** `False`
skewness	0.777725426986982	0.777725426986982
kurtosis	0.016046316181234515	0.016046316181234515

From the table above, P-value is 0.778 for skewness, no matter setting `bias` = `True` / `False`. We cannot reject the null hypothesis that the skewness is unbiased in 77.8% confidence. However, P-value is 0.016 for kurtosis, no matter setting `bias` = `True` / `False`. We have 98.4% confidence to reject the null hypothesis that the kurtosis is unbiased.

By simulation, the two kurtosis functions, no matter setting `bias` to `True` or `False` , are both biased; while the two skewness functions are both likely to be unbiased.

Problem 2

Problem

Fit the data in problem2.csv using OLS and calculate the error vector. Look at its distribution. How well does it fit the assumption of normally distributed errors?

Fit the data using MLE given the assumption of normality. Then fit the MLE using the assumption of a T distribution of the errors. Which is the best fit?

What are the fitted parameters of each and how do they compare? What does this tell us about the breaking of the normality assumption in regards to expected values in this case?

Answer

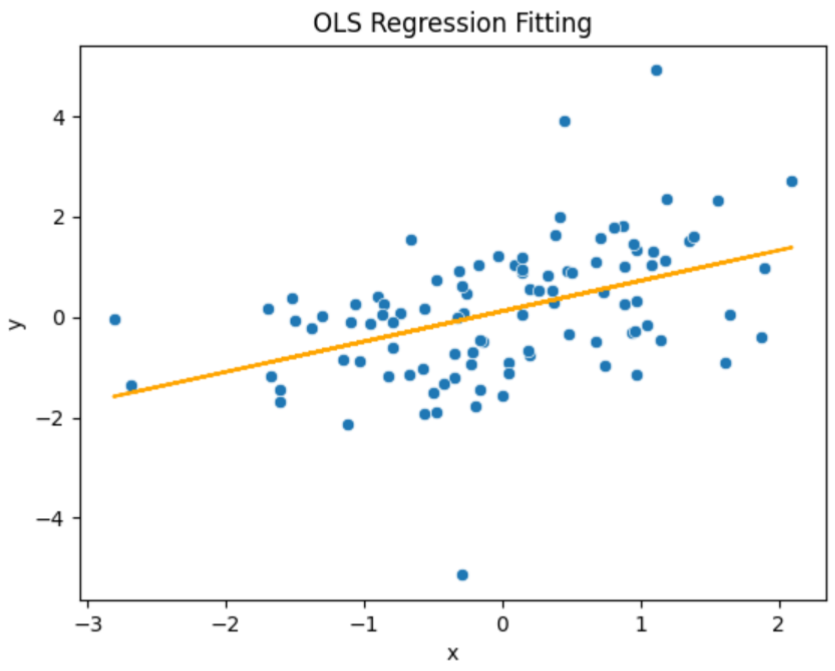
1. OLS fitting

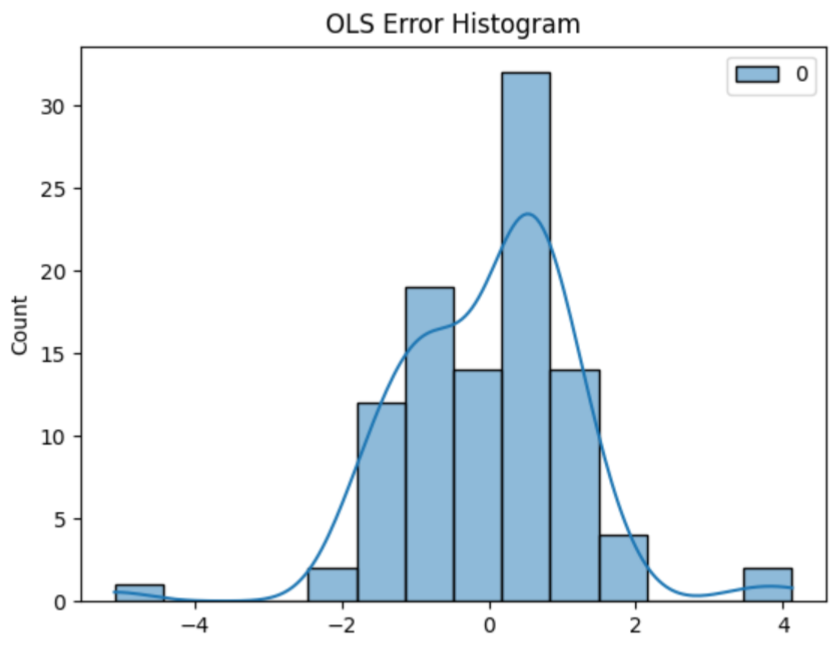
To do OLS fitting, I used the `LinearRegression` function in `Scikit-Learn`. The result is as follows.

$$y_{pred} = 0.60520482 * x + 0.1198362$$

$$R^2 = 0.19463952$$

The error vector can be calculated as $\epsilon = y - y_{pred}$. The following are OLS fitting plot and OLS error histogram.





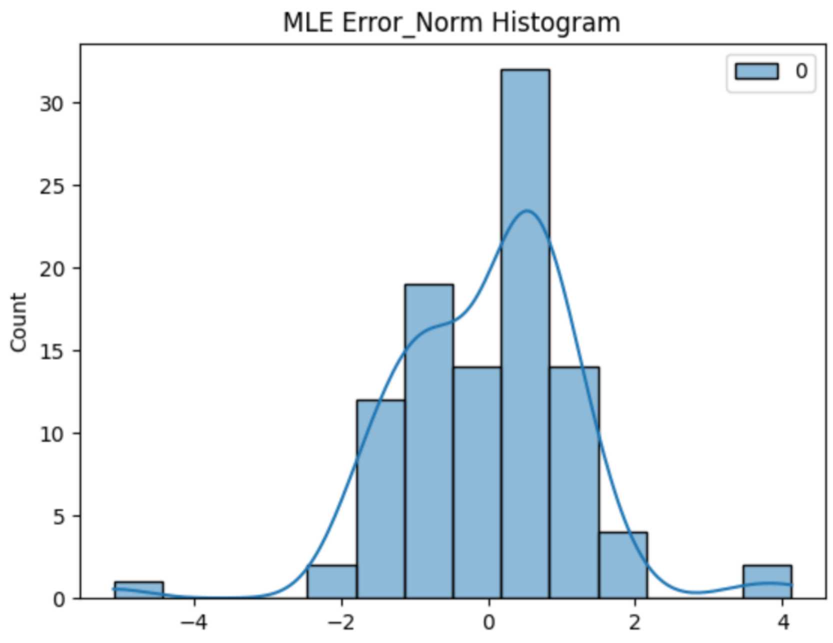
From the plots, we can see that the error distribution does not fit the assumption of normally distributed errors very well. The mean is a little bit larger than 0, and it is not a symmetric plot, with the skewness smaller than 0. Meanwhile, there are outliers near -4 and 4, in this way, the excess kurtosis is not 0 as well.

2. MLE fitting

First, fit the data using MLE given the assumption of normality. I calculated the log likelihood function and used the `minimize` function in `SciPy.stats` to maximize the log likelihood function under the assumption of normality. The result and error histogram are as follows.

$$y_{pred} = 0.60520485 * x + 0.1198362$$

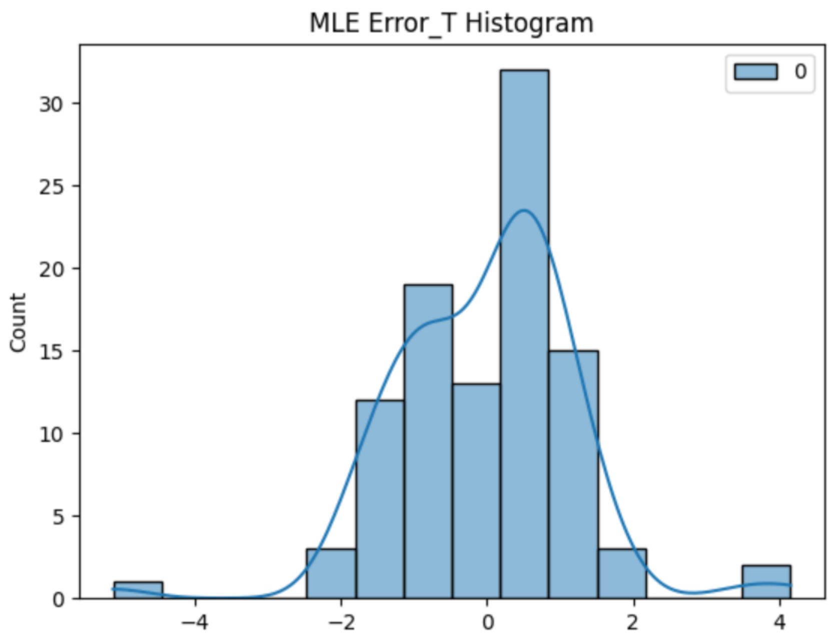
$$AIC = 325.98419 \ \ BIC = 333.79970 \ \ R^2 = 0.19463952$$



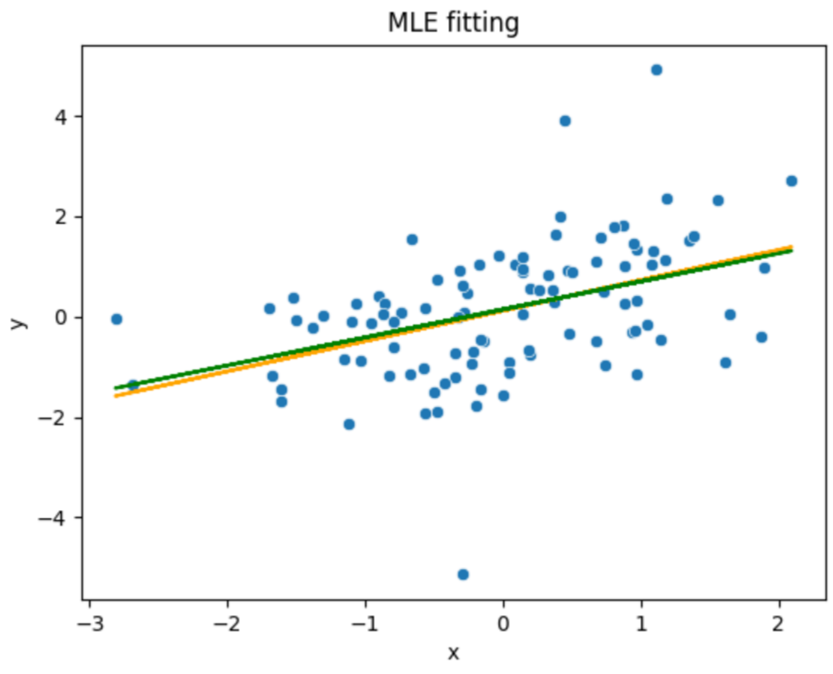
Then, fit the data using MLE given the assumption of T-distribution. I calculated the log likelihood function and used the `minimize` function in `SciPy.stats` to maximize the log likelihood function under the assumption of T-distribution. The result and error histogram are as follows.

$$y_{pred} = 0.5575717 * x + 0.1426142$$

$$AIC = 318.94594 \ \ BIC = 329.36662 \ \ R^2 = 0.19337752$$



The MLE fitting plot is as follow, with normal assumption the orange line and the T-distribution assumption the green one.



By comparison, the coefficient and intercept are very close, and R^2 is close, with the normal one a little bit larger. However, from the value of AIC and BIC, we know that T-distribution's is smaller, which is better. In the case of similar R^2, we use the AIC and BIC to conclude that the T-distribution assumption one is the better fit. Meanwhile, by looking at the error histograms, the assumption of normality is not likely to be true. To sum up, the T-distribution assumption one is the better fit.

3. Comparison of OLS and MLE

The fitted parameters and R^2 are shown above. By comparison, the fitting results of OLS regression and MLE estimation under assumption of normality are almost the same. And they are very close to the result of MLE estimation under assumption of T-distribution. Although the error of OLS regression and MLE estimation of this dataset breaks the assumption of normality, it does not actually affect the fitting results a lot. They can still provide a good fit, compared with the MLE estimation under T-distribution assumption.

Problem 3

Problem

Simulate AR(1) through AR(3) and MA(1) through MA(3) processes. Compare their ACF and PACF graphs. How do the graphs help us to identify the type and order of each process?

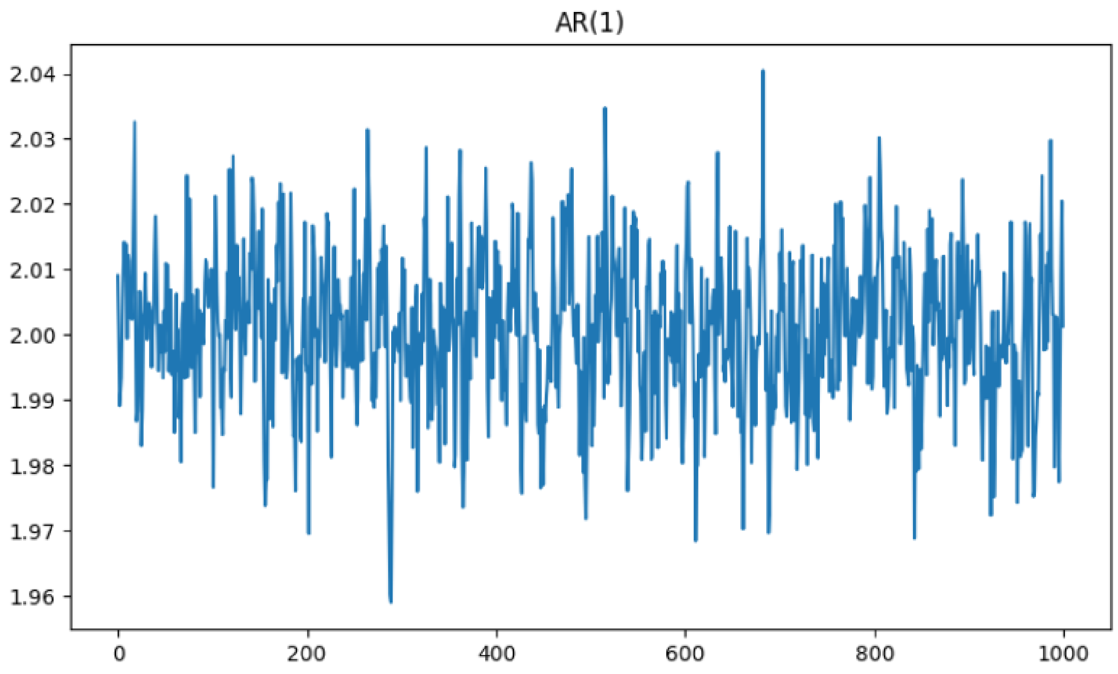
Answer

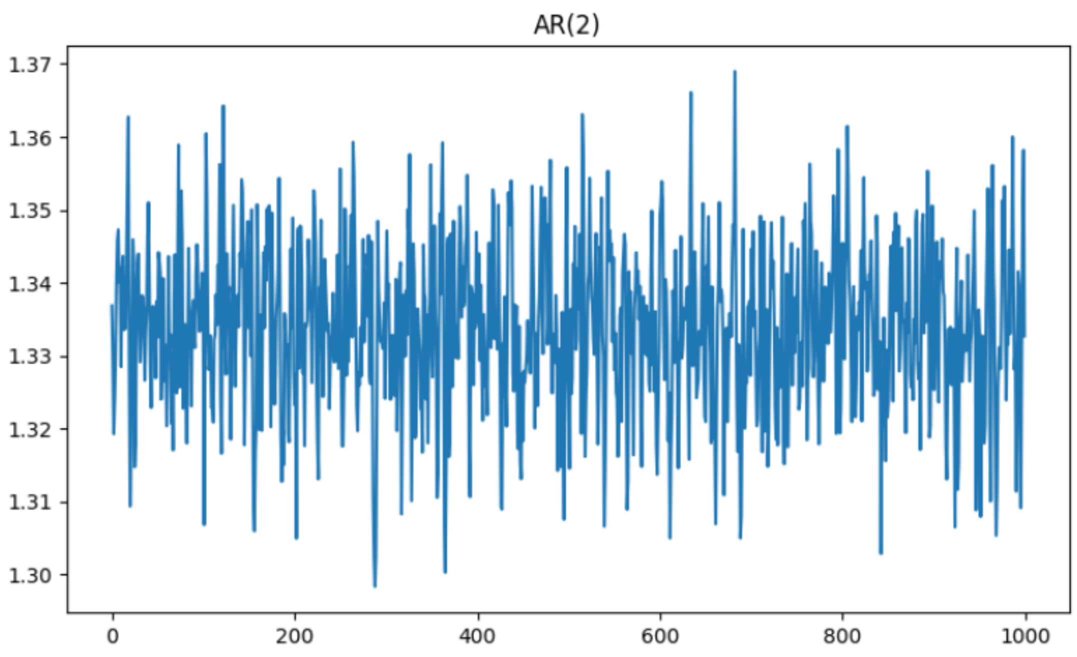
Simulate AR(1) through AR(3). Since $x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + \ldots + \phi_p x_{t-p} + w_t$, we need to set some $\phi_p < 0$, so that $\phi_p x_{t-p}$ will not be monotonously increasing. The simulation is as follows, where $\epsilon \sim \mathcal{N}(0, 0.01)$.

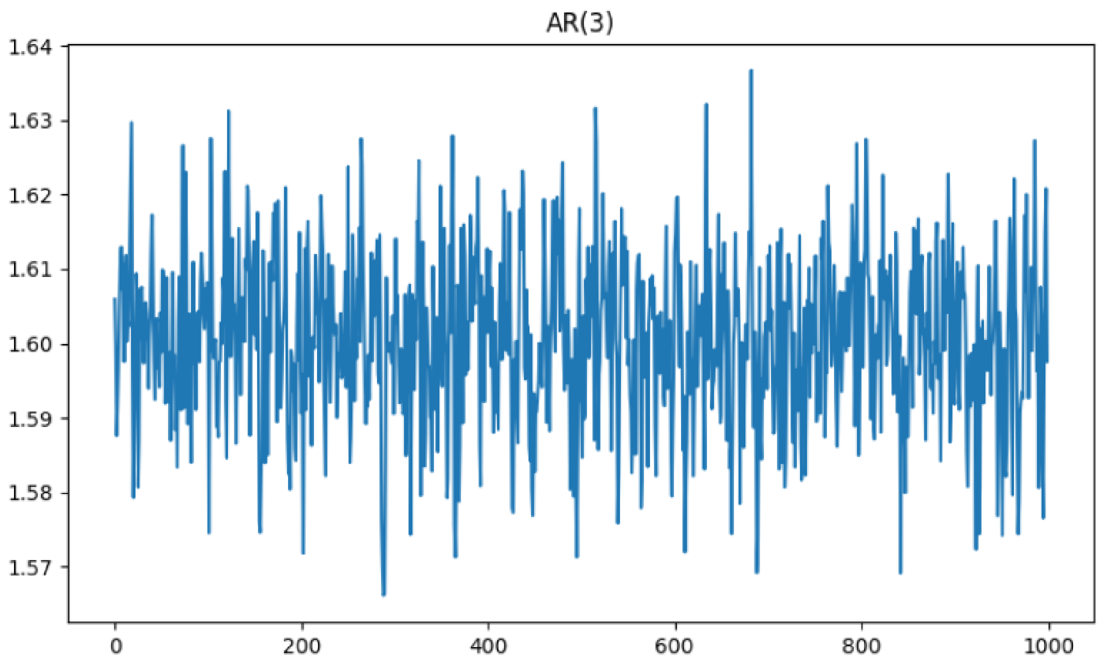
$$AR(1) : x_t = 1 + 0.5 \times x_{t-1} + \epsilon_t$$

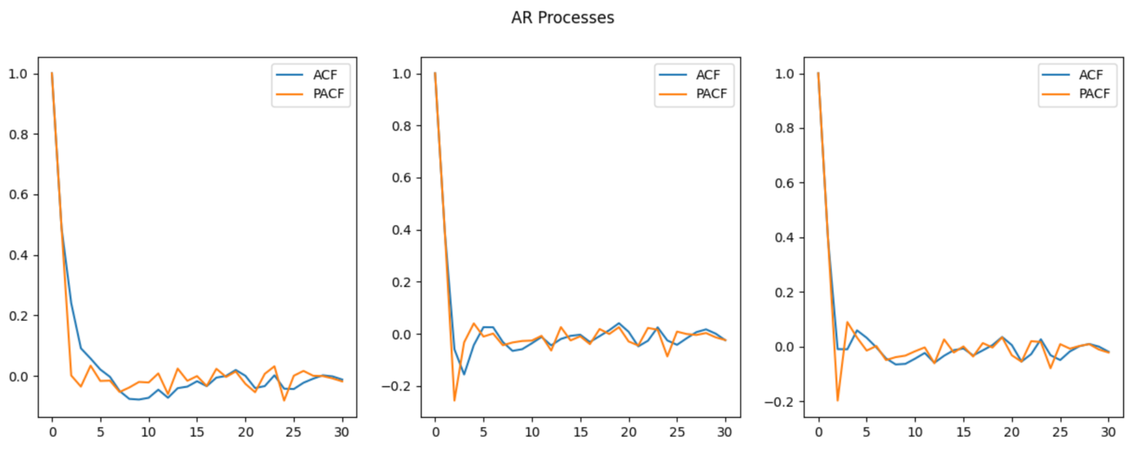
$$AR(2) : x_t = 1 + 0.5 \times x_{t-1} - 0.25 \times x_{t-2} + \epsilon_t$$

$$AR(3) : x_t = 1 + 0.5 \times x_{t-1} - 0.25 \times x_{t-2} + 0.125 \times x_{t-3} + \epsilon_t$$







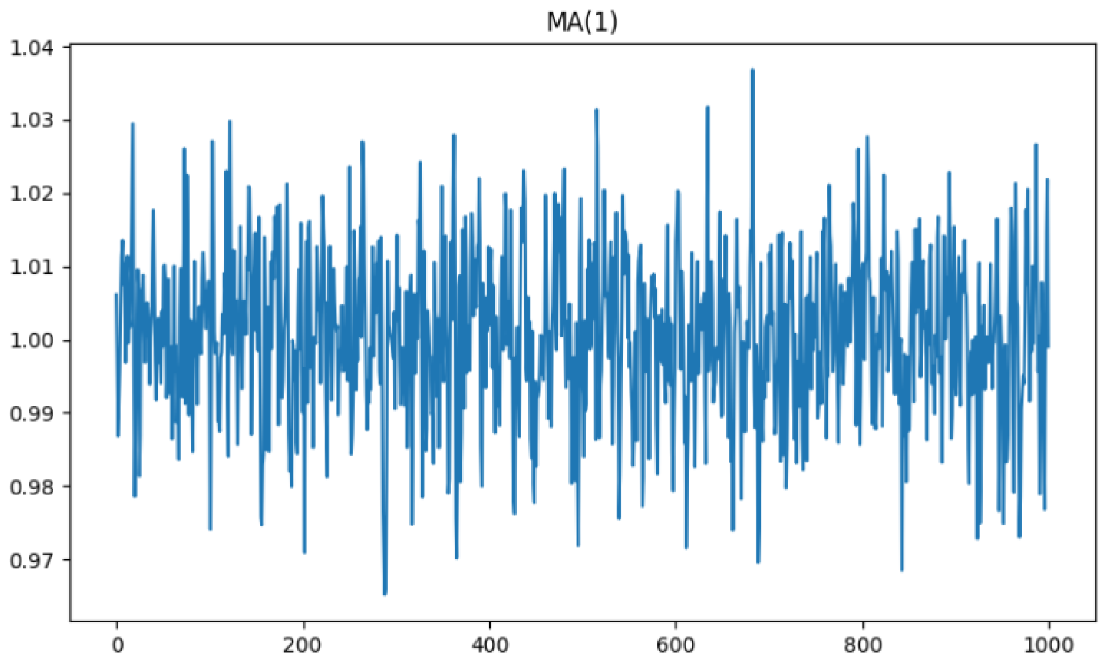


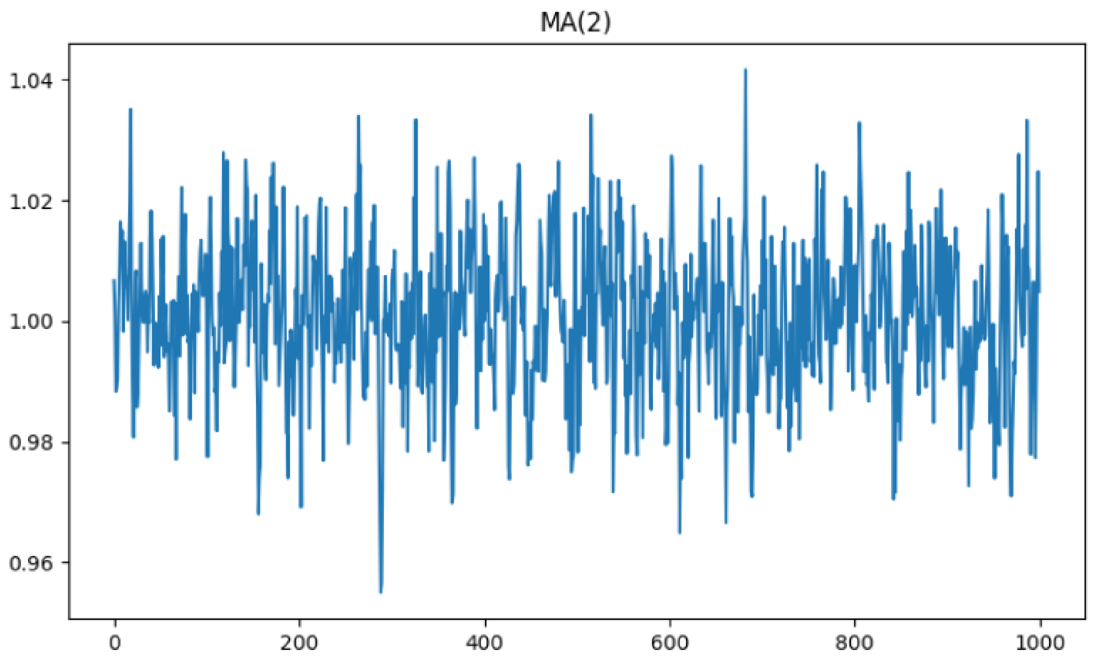
Simulate MA(1) through MA(3). Since $x_t = \theta_1 w_{t-1} + \theta_2 w_{t-2} + \ldots + \theta_p w_{t-p} + w_t$, the simulation is as follows, where $\epsilon \sim \mathcal{N}(0, 0.01)$.

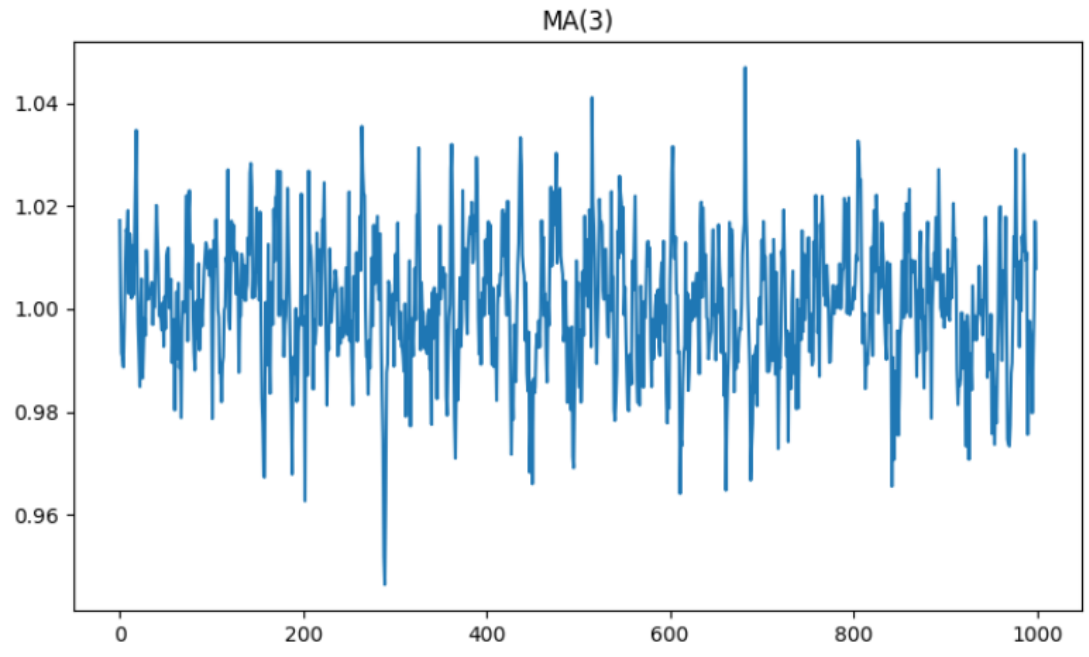
$$MA(1) : x_t = 1 + 0.5 \times \epsilon_{t-1} + \epsilon_t$$

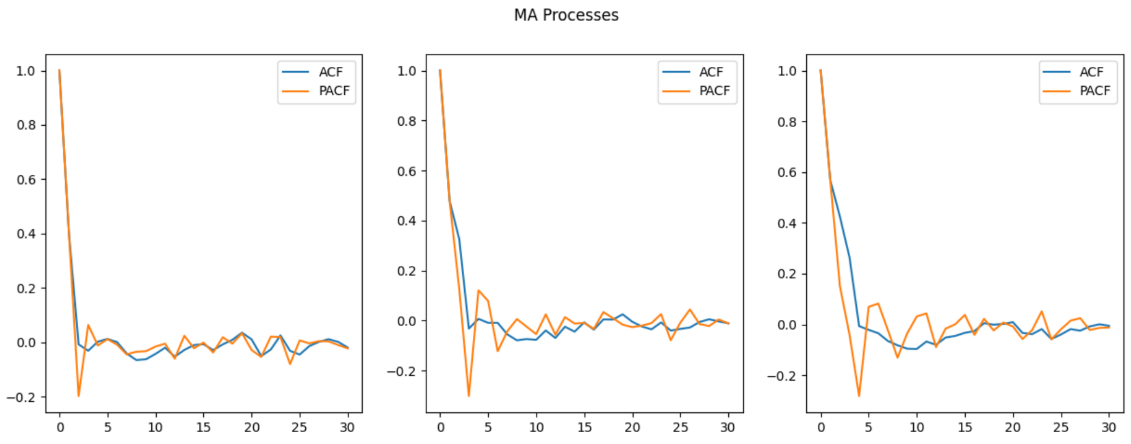
$$MA(2): x_t = 1 + 0.5 \times \epsilon_{t-1} + 0.5 \times \epsilon_{t-2} + \epsilon_t$$

$$MA(3): x_t = 1 + 0.5 \times \epsilon_{t-1} + 0.5 \times \epsilon_{t-2} + 0.5 \times \epsilon_{t-3} + \epsilon_t$$









Since we set some $\phi_p < 0$ for AR(p), as p grows, AR(p) will not change evidently. It almost maintains the same trend, but the fluctuation is a little more evident through AR(1) to AR(3). The shape of ACF and PACF for AR is also similar, maintaining the same trend and value, due to some negative ϕ_p. To sum up, AR(p) will change faster between x_t and x_{t-1}, because x_t is, to some extent, the accumulation of a part of x_1 to x_{t-1}, thus the value will change faster, and will be more easily influenced by the ϕ_p we choose. The order cannot be easily differentiated by the graphs because of ϕ_p.

However, the moving trends of MA(p) is more obvious. Although maintaining similar pattern as well, as p grows, the data value maintains almost the same, the fluctuation becomes obviously smaller, making the data series smoother. The fluctuation of ACF and PACF for MA also becomes smaller as p grows, with similar trends. MA(p) will not be influenced that much by the θ_p we choose, we can identify the order by the comparing the fluctuation of the graphs.