Problem 1:

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:
During the experiment devise phase, as we know, when sample size increases, the biased and unbiased parameters should converge to their true value, and we are interested in whether it is true, so we allow it to vary in sample_size. Also, as shown during class, with fewer sample, we have less statistical power to detect bias, even if it exists, so we also allow it to vary in sample_round. Lastly, we present null hypothesis for both kurtosis test and skewness test with threshold significance level being 5%, namely,
H0: the kurtosis function is unbiased in chosen statistical package,
H0: the skewness function is unbiased in chosen statistical package.
The statistical package used to generate kurtosis and skewness is scipy.stats.

Below is the table for kurtosis tests:

sample size	sample round	biased kurtosis	unbiased kurtosis	p-value	H0 result
100	100	1.974	0.034	1.69E-30	reject
	1000	9.923	-0.008	1	
1000	100	3.376	-0.003	1.29E-05	reject
	1000	3.585	1.112	2.32E-23	reject
100000	100	-0.002	3.377	0.983	
	1000	1.42E-05	1.31E+177	0.318	

As can be obtained from the table, when sample size is 100 and sample round is 100 / sample size is 1000 and sample round is 100 / sample size is 1000 and sample round is 1000, we rejected the null hypothesis and kurtosis is biased, while in other situations, we are unable to reject the null hypothesis and thus kurtosis can be unbiased. From our result, it is safe to conclude that when sample size is small, the kurtosis calculated by scipy.stats tend to be biased; and when we increase sample, the kurtosis will converge with the unbiased kurtosis. As for sample round, we do witnessed difference in H0 result for the same sample size under different sample rounds, which could be an indication of tests with fewer sample round failing to detect bias correctly. However, it could be also related to sample size or algorithm in scipy.stats, thus further research would be required to fully explain this issue.

Below is the table for skewness tests:

sample size	sample round	biased skewness	unbiased skewness	p-value	H0 result
10	10	0.019	0.0129	0.343	
	100	1.984	0.042	4.63E-31	reject
	1000	-0.0004	0.0006	0.318	
1000	10	-0.133	0.053	0.582	
	100	0.039	1.98	0.868	
	1000	0.488	0.0007	0.686	
100000	10	-0.003	0.674	0.993	
	100	1.981	0.039	6.61E-31	reject
	1000	0.0005	0.031	0.987	

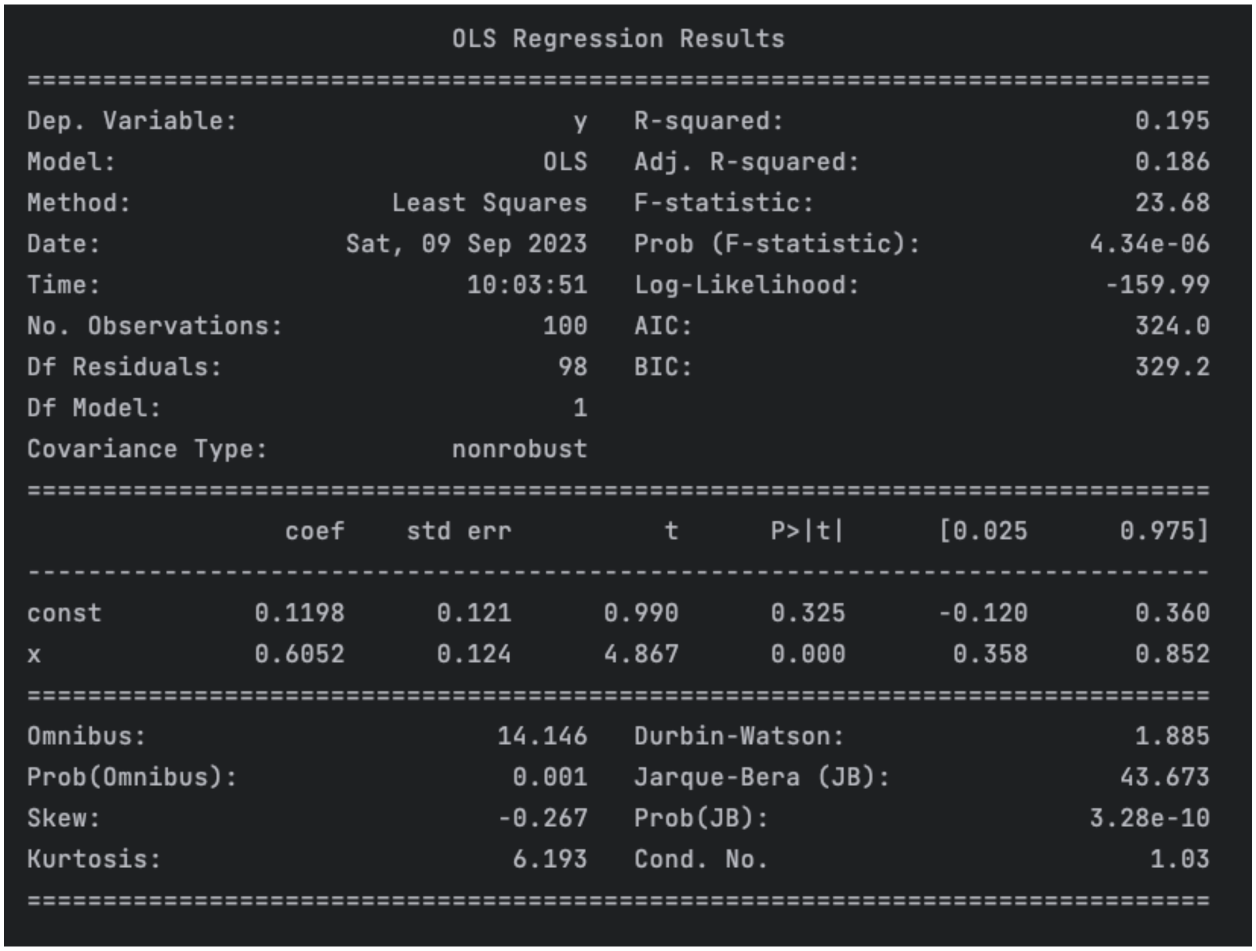
Due to the fact skewness tests shows much less rejections on hull hypothesis, sample_size = [100, 1000, 100000] and sample round = [100, 1000] would sometimes fail to reject null hypothesis during each test, so we expand the varying range to explore whether skewness function is indeed unbiased, or we are just not testing enough. As can be obtained from the table, when sample size is 10 and sample round is 100 / sample size is 100000 and sample round is 100, we rejected the null hypothesis and skewness is biased, while in other situations, we are unable to reject the null hypothesis and thus skewness can be unbiased. In general, we can still conclude that when we increase sample size and sample round, skewness calculate would tend to converge with unbiased parameter, as there is one rejection when sample size is 10, and none when sample size is 1000. However, this pattern is less obvious for skewness than for kurtosis, as one rejection can still be observed when sample size is 100000. This could be due to the fact that fewer sample round (100) failed to correctly capture the bias, as when we increased sample round (1000), such rejection is untenable, or skewness is intrinsically tend to be less affected by sample size, since higher-order moments tend to be more sensitive to sample size or outliers, and skewness only measures third central moment, which is fewer than that of kurtosis.

In conclusion, both kurtosis function and skewness function are biased in scipy.stats. Such biasness is more obvious for both kurtosis and skewness when sample size is small, and gradually disappears when sample size is increased and biased parameters converge with unbiased ones. However, it is worthy to note that such pattern is more obvious for kurtosis, while it is possible for skewness to (seem to) be unbiased even sample size is smaller. Such could be caused by the algorithm used in scipy.stats to calculate skewness, rounding errors during hypothesis test, or the fact that skewness is indeed less affected by sample size due to lower order than kurtosis.

Problem 2:

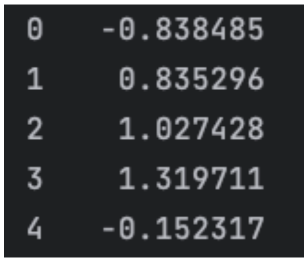
Fit the data in problem2.csv using OLS and calculate the error vector. Look at its distribution. How well does it fit the assumption of normally distributed errors? Fit the data using MLE given the assumption of normality. Then fit the MLE using the assumption of a T distribution of the errors. Which 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:
Below is the OLS fitting result:



```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.195
Model:                            OLS   Adj. R-squared:                  0.186
Method:                 Least Squares   F-statistic:                     23.68
Date:                Sat, 09 Sep 2023   Prob (F-statistic):           4.34e-06
Time:                        10:03:51   Log-Likelihood:                -159.99
No. Observations:                 100   AIC:                             324.0
Df Residuals:                      98   BIC:                             329.2
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.1198      0.121      0.990      0.325      -0.120       0.360
x              0.6052      0.124      4.867      0.000       0.358       0.852
==============================================================================
Omnibus:                       14.146   Durbin-Watson:                   1.885
Prob(Omnibus):                  0.001   Jarque-Bera (JB):               43.673
Skew:                          -0.267   Prob(JB):                     3.28e-10
Kurtosis:                       6.193   Cond. No.                         1.03
==============================================================================
```

Below is the error vector (partial):

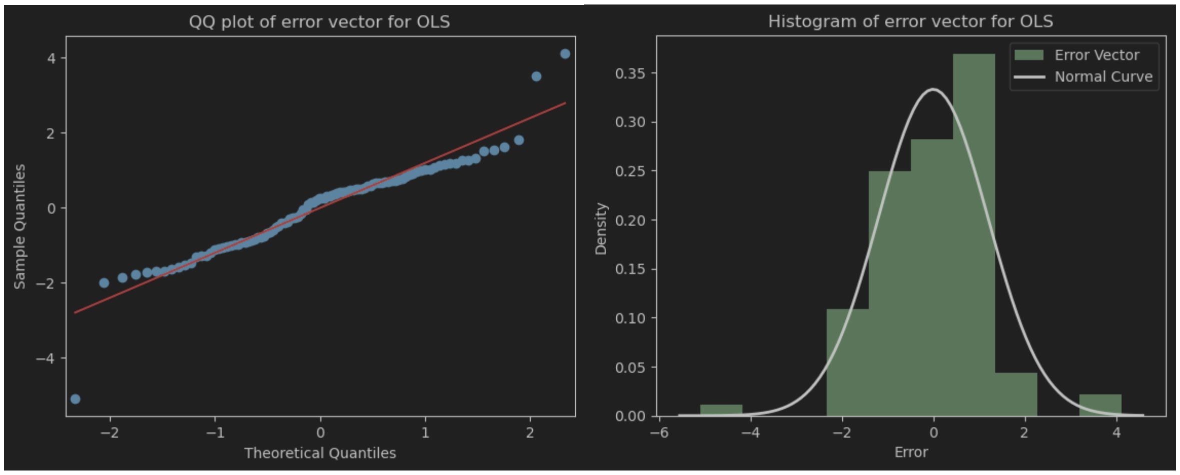


```
0   -0.838485
1    0.835296
2    1.027428
3    1.319711
4   -0.152317
```

Below is the parameter for error vector obtained through OLS:

mean	-3.89E-17
variance	1.436
skewness	-0.267
kurtosis	3.193

As can be obtained from the chart, if the residual were to be normally distributed, its skewness and kurtosis should be 0 (scipy.stats for specific), however its skewness is -0.267 and its kurtosis is 3.193. Such argument can also be backed by the QQ plot for its residual, as can be shown below. Admittedly, a large portion of errors do lie on the normal distribution line, but several outliers on both sides of the graph are not negligible. The comparison of residual's histogram and normal distribution curve also shows that the error vector is not strictly normal distributed and a bit left skewed. Thus, we could conclude that it does not fit the assumption of normally distributed errors very well.



We also fit the data using MLE under the assumption of normally distributed errors and T-distributed errors. The fitting result along with the parameters for assessing the goodness of fit are also presented in the table below.

	OLS	MLE_normal	MLE_T
beta0	0.1198	0.1198	0.1426
beta1	0.6052	0.6052	0.5575
R squared	0.195	0.195	0.193
adjusted R squared	0.186	0.186	0.185
AIC	323.98	323.98	314.95
BIC	329.19	329.19	320.156

We also performed error vector analysis for MLE under normal assumption and MLE under T assumption, as shown below. First, we can see from non-zero skewness and kurtosis for MLE under normal assumption that normal distribution of errors also doesn't hold here. Second, we can also see abs(skewness) for MLE under T assumption is smaller than that for MLE under normal assumption, but kurtosis for MLE under T assumption is bigger.

	MLE_normal residual	**MLE_T residual**
mean	4.90E-09	-0.023
variance	1.436	1.438
skewness	-0.267	-0.243
kurtosis	3.193	3.285

As for the fitting result table, for coefficients, beta0 and beta1 for OLS and MLE_normal are almost identical, while those of MLE_T differs from the former two, being 0.1426 for intercept and 0.5575 for slope. For the goodness of fit, we selected R squared, adjusted R squared, AIC and BIC. For R squared and adjusted R squared, higher value usually indicates a better fit; while for AIC and BIC, lower are preferred. As can be obtained from the graph, the R squared, adjusted R square, AIC and BIC are almost identical for OLS and MEL_normal, of which both are under the assumption of normal distribution. This means these two methods achieve similar level of goodness of fit. For MLE_T, R squared and adjusted R squared are around the same level as those of OLS and MLE_normal, but AIC and BIC are much lower than the former two. This indicates MLE fit under the assumption of T distributed errors is a better fit.

In fact, we could probably safely assert that a large portion of real-life data does not follow normal distribution and in many circumstances a perfect linear relationship does not exist between the regressors. Through this example, by changing the distribution assumption (from normal distribution to T-distribution), we may be able to get better fit.

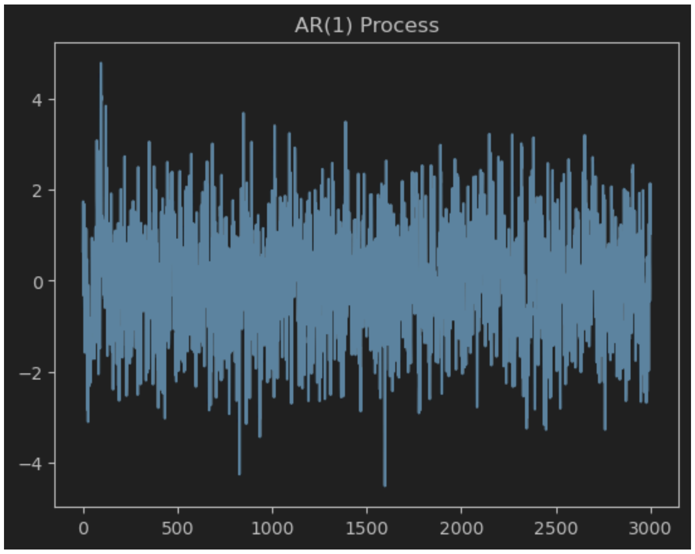
Problem 3:

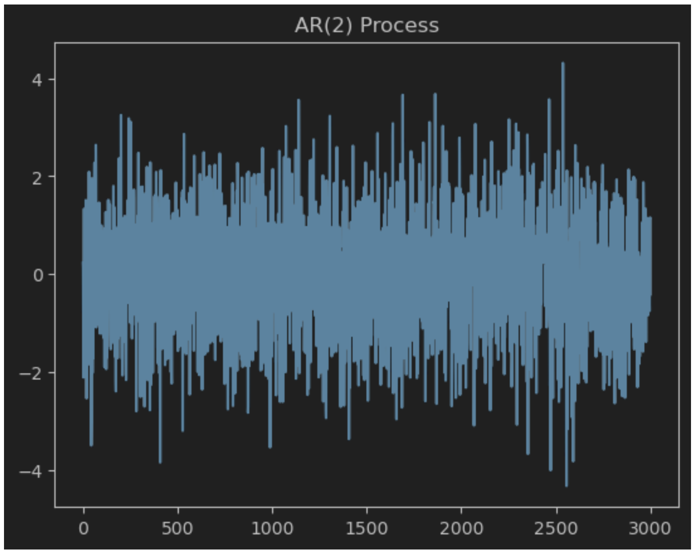
Simulate AR(1) through AR(3) and MA(1) through MA(3) processes. Compare their ACF and PACF graphs. How do the graphs help us to identify the type and order of each process?

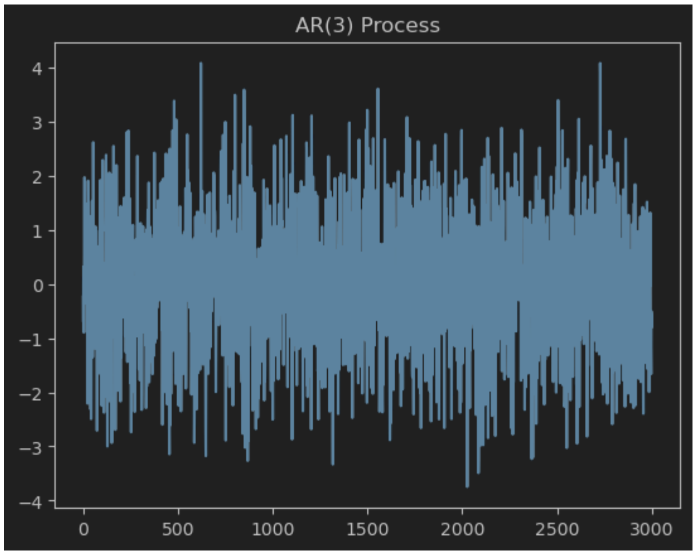
Answer:
It worth notifying that we applied sm.tsa.arma_generate_sample to simulate, this ARMA package allows us to simulate both AR and MA by using one function. By setting ma = [1], we can get pure AR simulations; and by setting ar = [1], we can get pure MA simulations.

ACF (autocorrelation function) describes the correlation between a series and its lags. In other words, it helps to understand how a given observation is related to its previous observations. PACF (partial autocorrelation function) describes the correlation between a series and its lags that is not explained by previous lags.

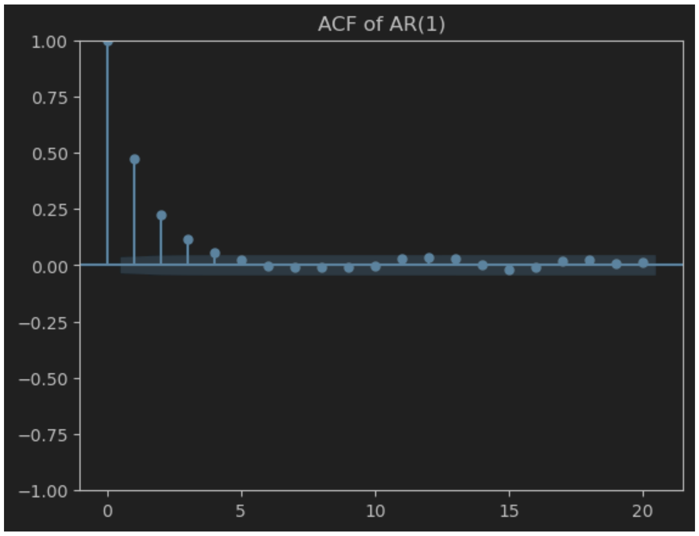
Below are the AR(1), AR(2), AR(3) simulations.

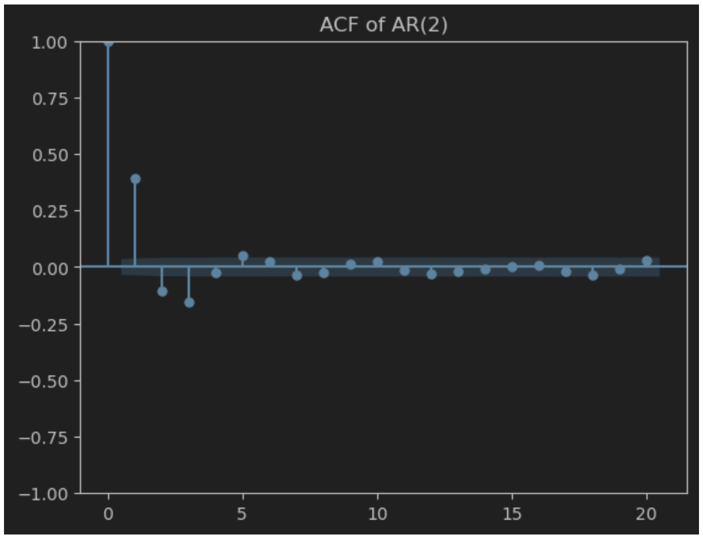


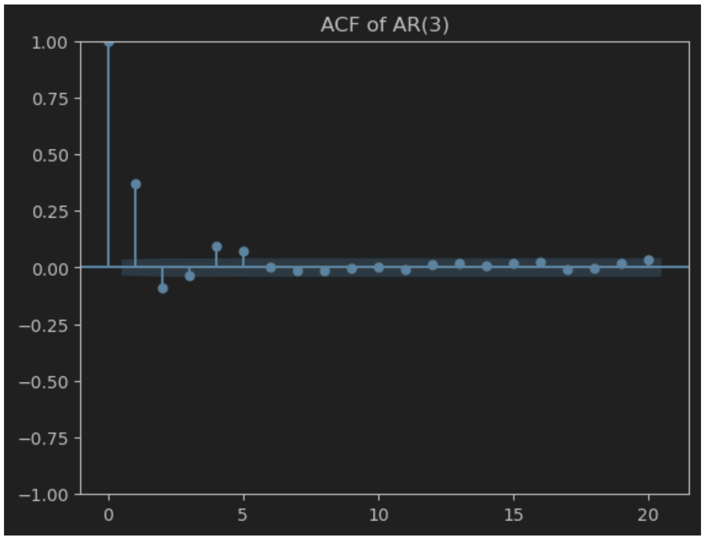




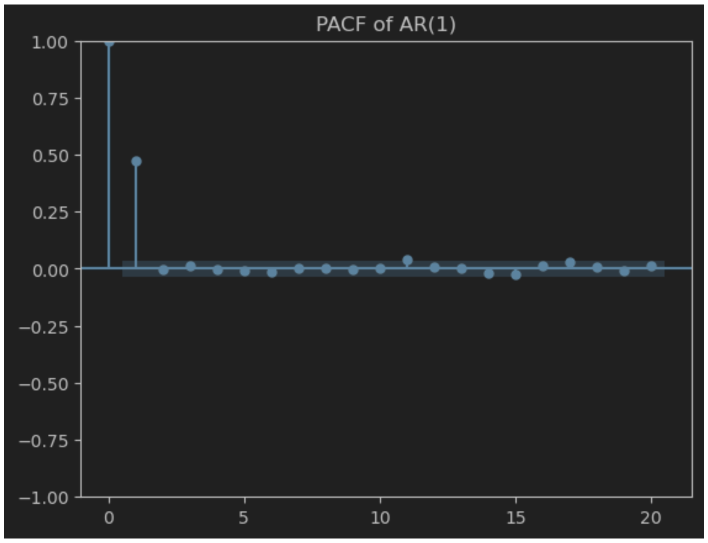
Below are the ACF for AR(1), AR(2), AR(3). A gradual decreasing trend can be witnessed as in AR current and lag terms are closely related.

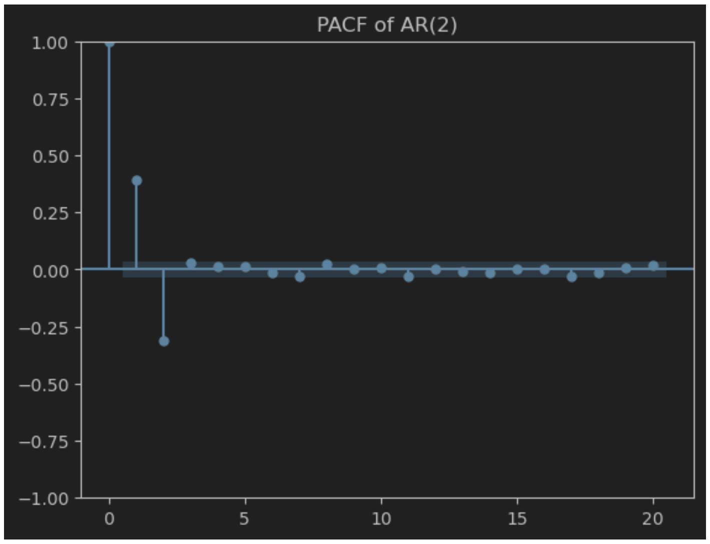


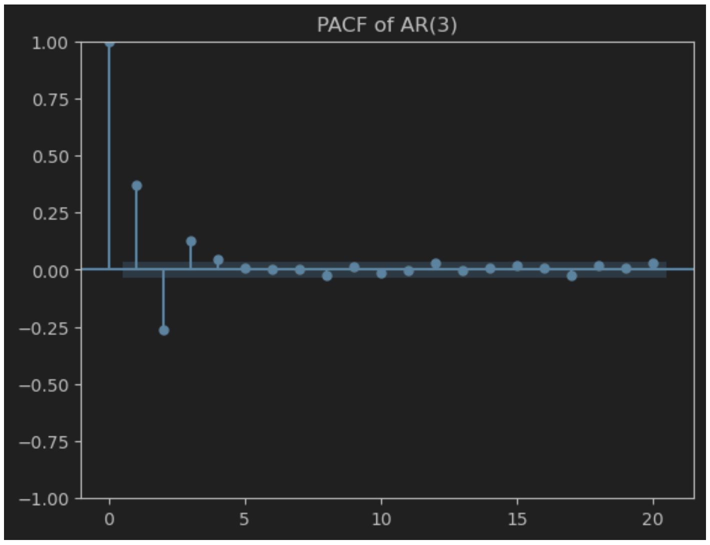




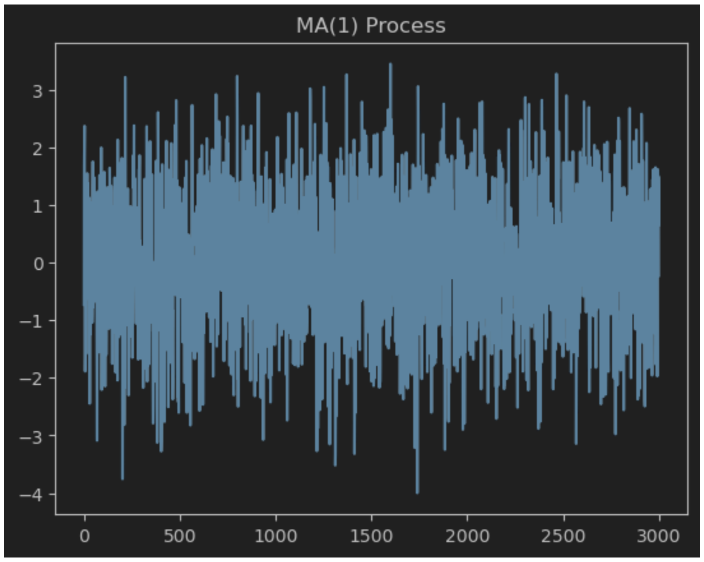
Below are PACF for AR(1), AR(2), AR(3). PACF for AR can be used to identify orders, the point where the PACF values become insignificant suggests the order of the AR model. It is obvious that for PACF of AR, there is a sharp cut-off towards 0 after a certain lag, as according to the formulas of AR, only those lags closer to the current term captures the change well. Thus, we can obtain from the graphs that AR(1) has 1 outstanding point (1 lag), AR(2) has 2 (2 lag), and AR(3) has 3 (3 lag). Such sharp cut-off in PACF can also be used to identify whether AR model is a better fit for the data given, and in our examples it indeed is.

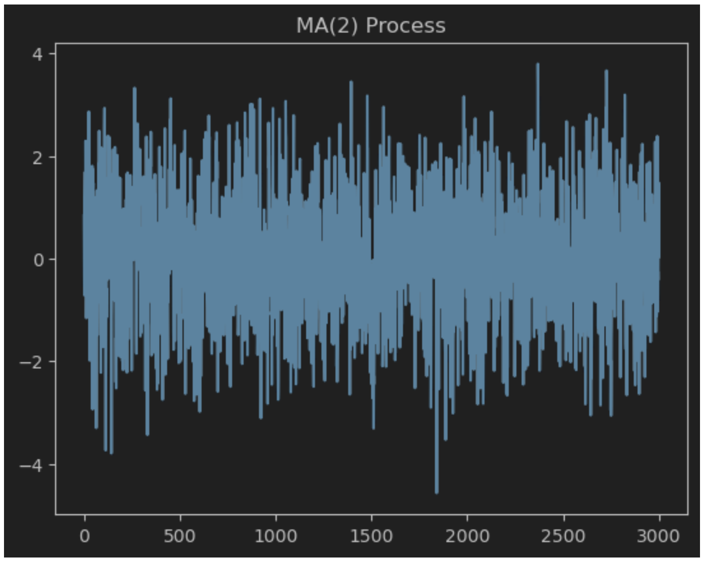

PACF of AR(1)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20

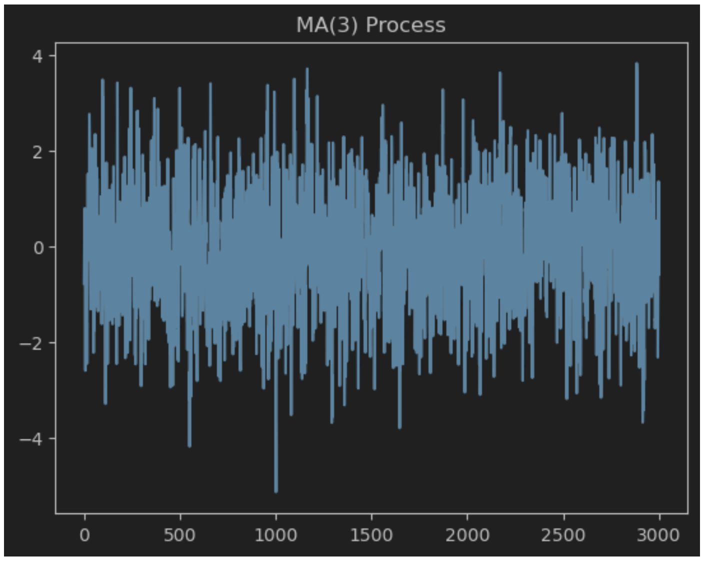

PACF of AR(2)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20


PACF of AR(3)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20

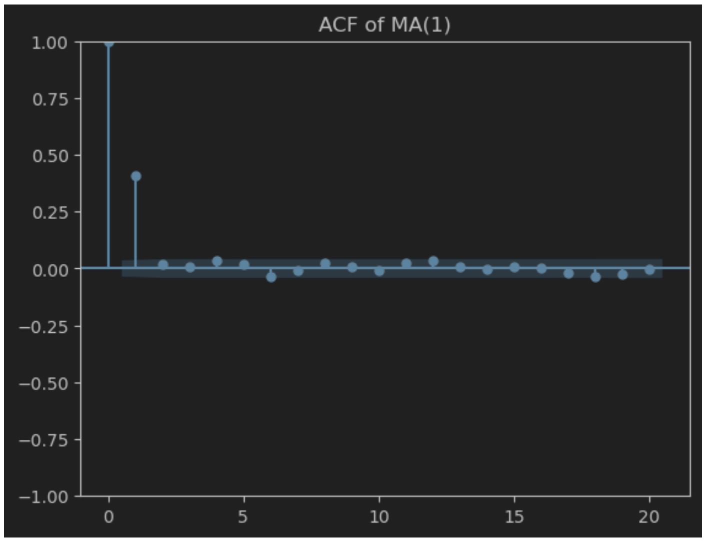
Below are MA(1), MA(2), MA(3) simulations.

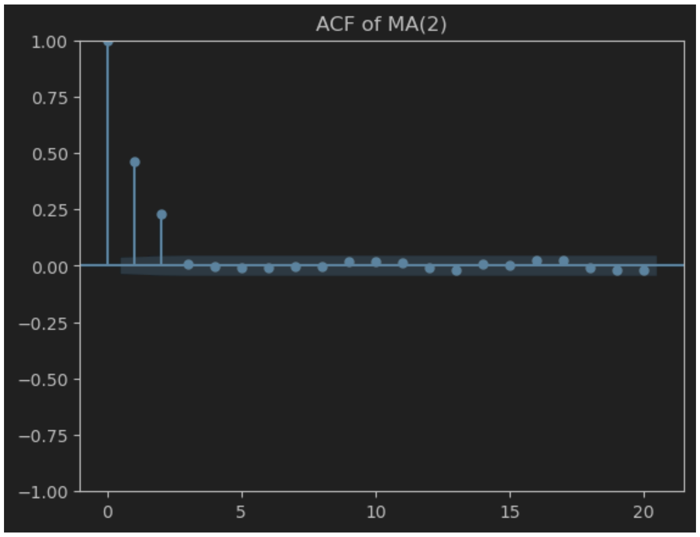


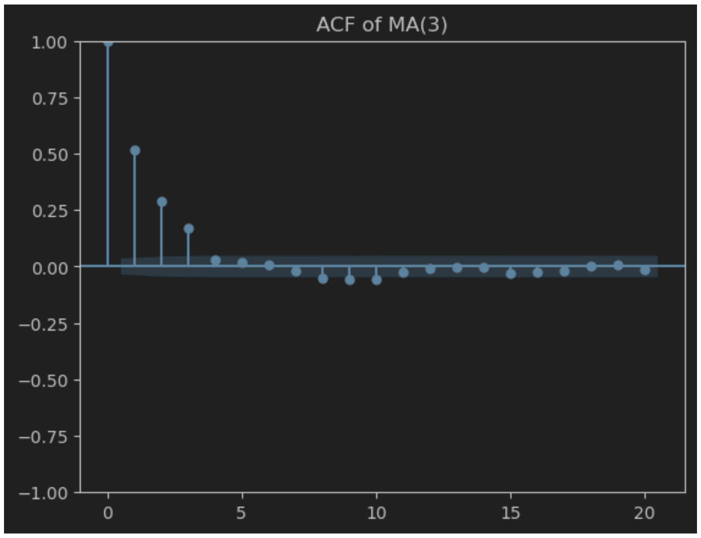




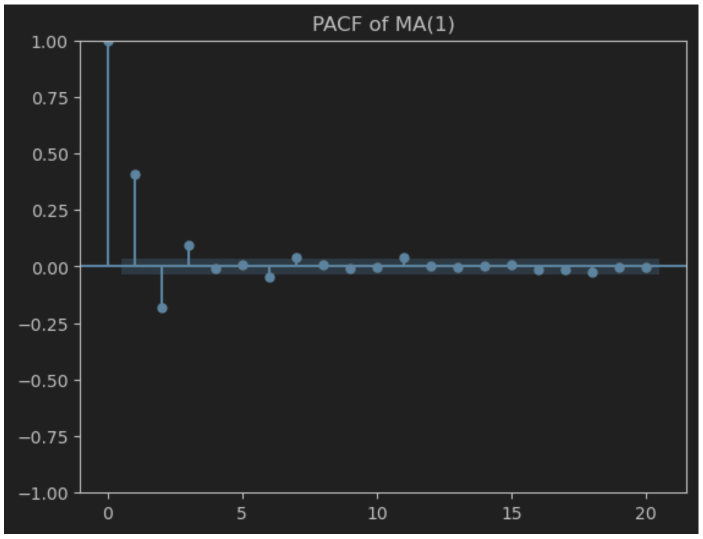
Below are ACF for MA(1), MA(2), MA(3). ACF for MA can be used to identify orders, if ACF cuts off towards 0 and becomes insignificant after a certain point, it is suggesting the order of MA model. Thus, we can obtain from the graphs that ACF of MA(1) becomes insignificant after 1st lag, therefore the model should have an order of 1; ACF of MA(2) becomes insignificant after 2nd lag, therefore the model should have an order of 2; ACF of MA(3) becomes insignificant after 3rd lag, therefore the model should have an order of 3.

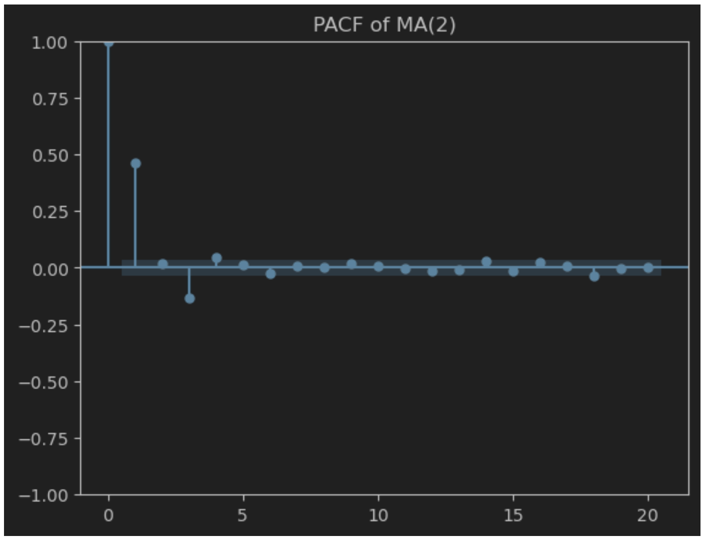


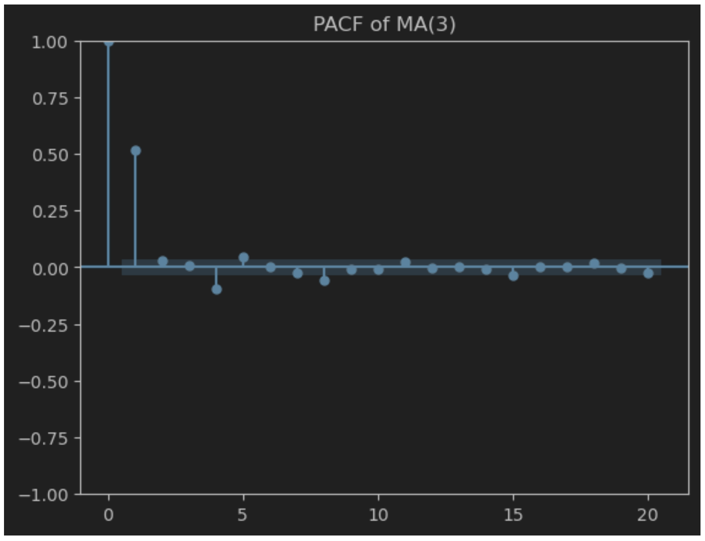




Below are PACF for MA(1), MA(2), MA(3). The graphs decay more slowly, and gradual decreasing trend can be witnessed, which also indicates they are MA models, as white noise error term is continuously exerts its effect on the model.


PACF of MA(1)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20


PACF of MA(2)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20


PACF of MA(3)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20

In conclusion, we would obtain more information in PACF if AR model is a better fit given the data, as each value in the series is a linear combination of its p previous values plus a white noise error term. The point in PACF where point is becoming insignificant can be used to determine the order of the model (pth lags). On the contrary, we would obtain more information in ACF if MA model is a better fit given the data, as each value in the series is a linear combination of the previous q white noise error terms. The point in ACF where point is becoming insignificant can be used to determine the order of the model (qth lag). PACF can also be applied to determine whether it is MA model: if PACF decays slowly, then it should probably be MA model.