Project Week02: Leslie Dees

Problem 1:

Is your kurtosis function biased? Prove or disprove your hypothesis.

The kurtosis function that I used was based on the scipy.stats.moment method. This was used for calculations of the mean, variance, skew, and kurtosis. Within this calculation I needed to consider the assumption of normality. This resulted in the final produced kurtosis from my function to be calculated with a subtraction of 3 in order to find the excess kurtosis. I added an input variable as well for testing that produced the exact kurtosis, which allows for simple comparisons against a standard normal.

In order to test if this moment method was biased, I decided to perform a series of kurtosis estimations on a known distribution. I chose a normal distribution for sampling; these distributions are known to have a kurtosis of 3. Therefore, my goal is to ensure that the kurtosis that I was estimating is 3. For sampling I chose to test over 5000 samples each with a sample size of 5000. This ensured a robust enough sample size to average over avoiding outliers while also having a large enough quantity of samples within tested distributions to avoid internal outliers. Samples were drawn from a normal distribution N~(0, 1). I calculated the kurtosis for each internal sample and then averaged the kurtosis against all of the sample runs.

The final results are as follows:

Known kurtosis: ***3***

Average Estimated Kurtosis (Sample Size 5000): ***3.0010035***

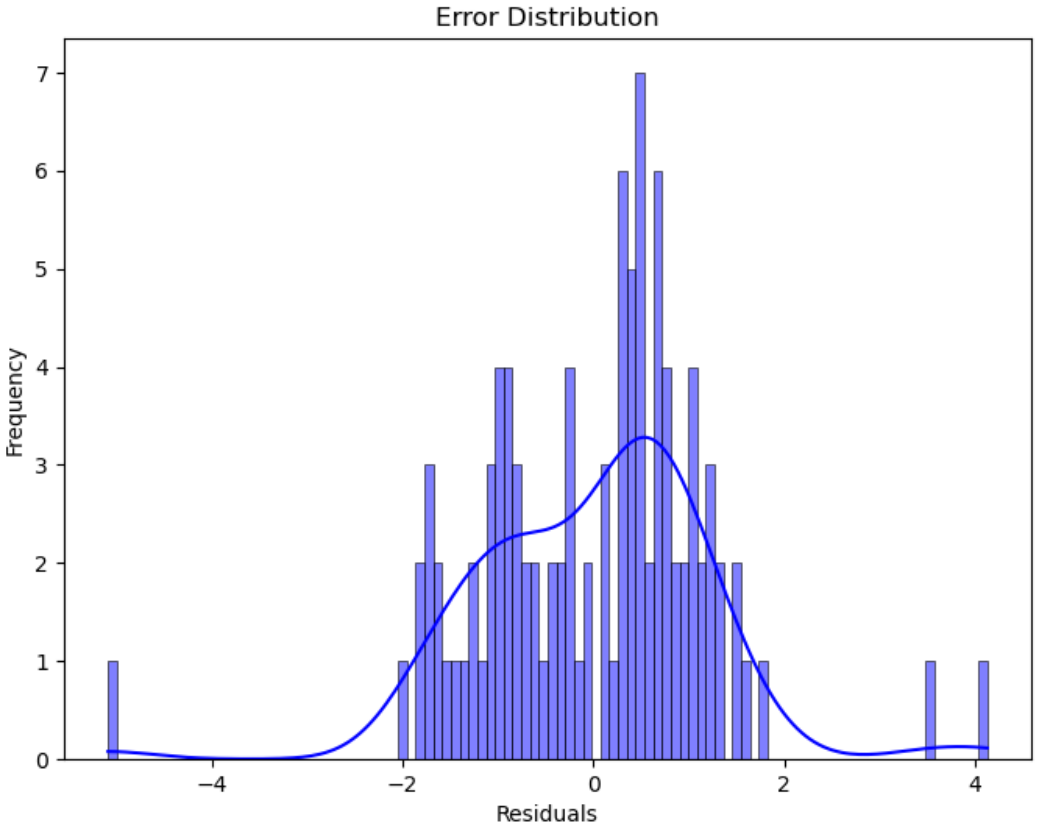
P-value: ***0.6122148***

As observed from the above calculation, the average estimated kurtosis is extremely similar to the known kurtosis over 5000 trials. This alone was not enough to validate my hypothesis that kurtosis is unbiased. I performed a hypothesis test which produced a p-value of 0.6122148 against the null hypothesis that kurtosis is unbiased. I fail to reject this hypothesis due to $p > 0.05$ and can therefore statistically say that the kurtosis function in this case is unbiased.

Problem 2:

Fit the data in problem2.csv using OLS and calculate the error vector. Looking at the distribution, how well does the fit assumption of normally distributed errors?

I formulated my OLS method using the statsmodels.api.sm package in Python. This method allowed for simple application of the OLS and easy observation of the error vector. The final error vector is too long to include in this writeup, but instructions on obtaining it are included in the README of p2_work.py. The error vector was calculated by obtaining the model residuals.



From observing the plot of the error vector of OLS we can see that it approximates a normal curve with slight skewing. However, I also obtained an average of the error vectors and the variance.

Averaged Error Vector: ***-4.77395e-17***

Variance Error Vector: ***1.43615***

This averaged error vector is approximately 0, which tells me that the OLS fit assumption is quite good for normally distributed errors. While there is some level of skew to the error vector distribution plot, I can assume that it approximates a normal curve with differences in variance rather than the mean. The variance is significantly different than a standard normal curve though. All of this taken into account, I make the deduction that the errors are approximately normally distributed.

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

I manually formulated my MLE for normal distribution assumptions manually using the equation for normal distribution log likelihood. However, for the t distribution of errors, I decided to use the scipy.stats.t.logpdf to assist with my calculation. Both of the methods used scipy.optimize.minimize to optimize the MLE. From the calculations, I performed a two-tailed hypothesis test to reject the null hypothesis that the optimized distributions were approximate to a normal and a t-distribution respectively.

Distribution Assumption	Normal	T
P-value	0.99991	0.93227
Test Statistic	-0.00011	-0.08507

By looking at the P-values of the distribution optimizations alone I can make the observation that the assumption of normality is a better fit. The normality assumption has a higher probability of the observed distribution fitting a normal distribution than the t distribution assumption.

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 regard to expected values in this case?

As for the fitted parameters, the optimized mean and standard deviation were calculated for both, while I attempted to optimize degrees of freedom as well for the t distribution case.

Distribution Assumption	Normal	T
Mean	-0.00011	-24.62520
Variance	0.97346	4793.37184
Degrees of Freedom	N/A	274.21126

It is notable that the mean and variance are very easily compared to a standard normal distribution, which this data appears to follow. As for the t distribution, the primary parameter obtained is the degrees of freedom. I initialized the degrees of freedom to N-1, where N is the number of samples. What we can see is that from the quantity of training that I did, the degrees of freedom drastically increased from this initial assumption. As for the mean and variance, it is difficult to compare since the t distribution primarily relies on the degrees of freedom.

We can see from all of these parameters though that breaking the normality assumption in regard to expected values results in wild results. The mean of the t distribution does not result in a 0-centered distribution and its variance implies a very wide spread of data. The high degrees of freedom implies that the t distribution requires a higher degree of freedom to match with the normally distributed data. These differences imply that while a t-distribution can fit the data, the normally distributed assumption fits much better.

The primary implication is that the normality assumption is not broken by this case. The expected values described by the assumption of normality fit a normal distribution very well. The better fit of p-value for the normality assumption also helps to prove that normality is not broken.

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 identify the type and order of each process?

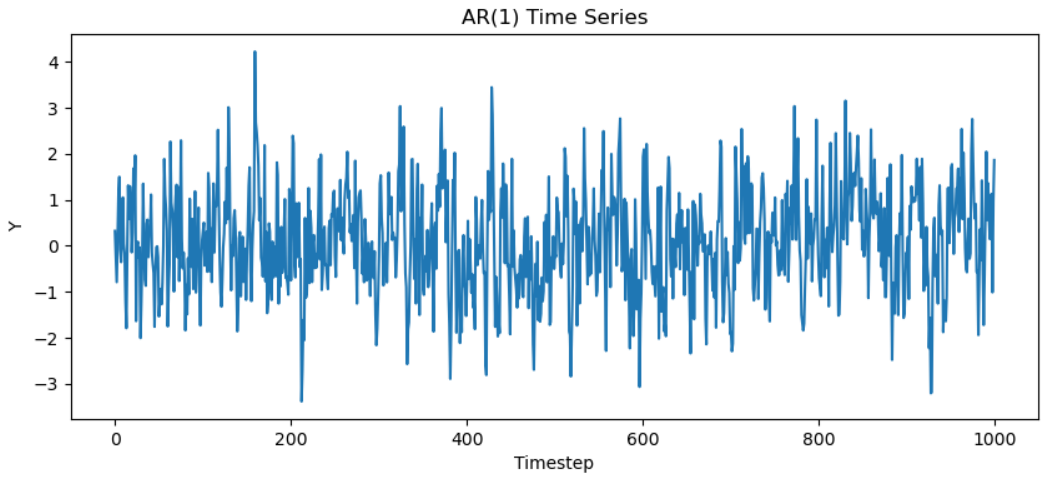
I used a common set of data obtained from a random normal distribution N~(0,1) using the same seed for each of the following trials. Moving average and Autoregression methods were manually calculated and available within fin_package.py. I plotted time series predictions for all AR and MA processes for N = 1, 2, 3 as well as the ACF and PACF graphs all described below. For the sake of readabilility, I will answer the question prior to showing the plots.

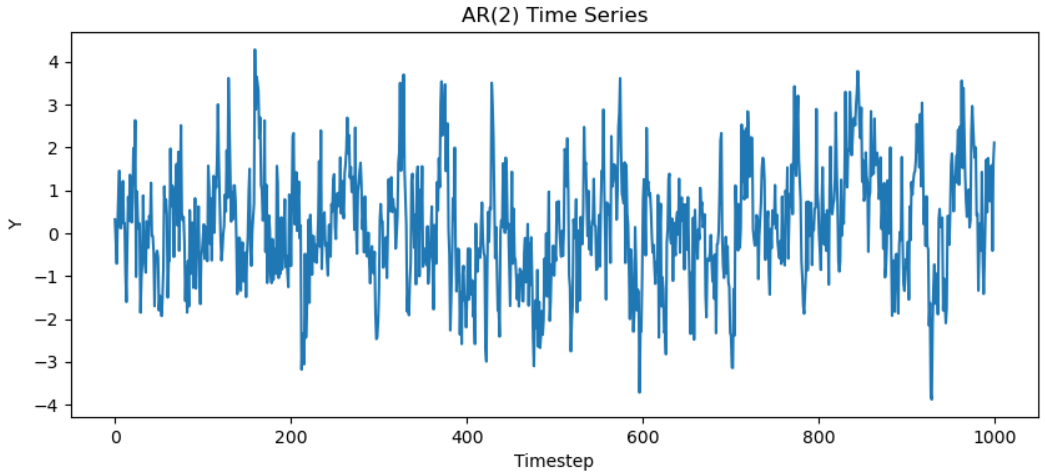
In order to interpret the type and order of each process I look into what lag the spikes occur on and the sharpness of the decline in consecutive lags. For existing consecutive lags, I must consider the steepness of the drop and the height of the 2nd lag in order to determine if it is significant enough to make me conclude in a higher-ordered process.

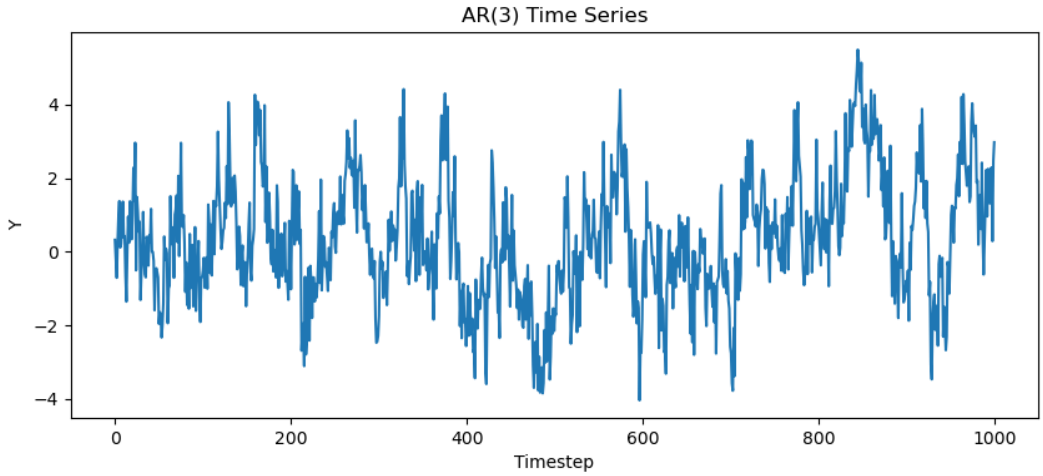
As depicted in the MA PACF and ACF plots, we can see that there is really only a large spike at lag 1. This means that we can assume that all of the higher order autocorrelations can be represented by only using a lag of 1. This is consitent through all of the MA(1) thru MA(3) ACF and PACF plots. Therefore I make the deduction that this is a MA(1) process.

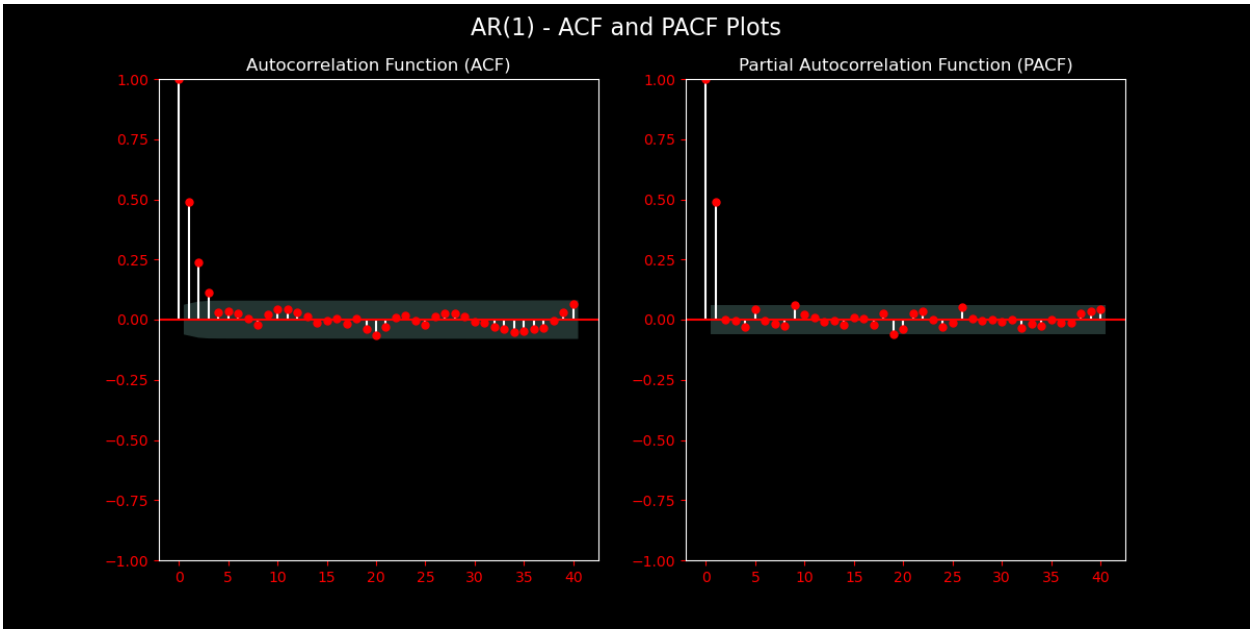
In the AR plots I can see that there appear to be multiple spikes in the PACF plot. While there is still a significant spike in the lag 1, I also see a large spike in lag 2. In the AR(3) plot I do see some spikes in the 3rd and 4th lags, but they are not as significant as the 1st and 2nd order lags. The ACF plots are harder to interpret for the AR process, however the AR(1) plot does hint at an AR(2) process due to a steeper decline after the lag 2. From these interpretations I deduce that this is an AR(2) process.

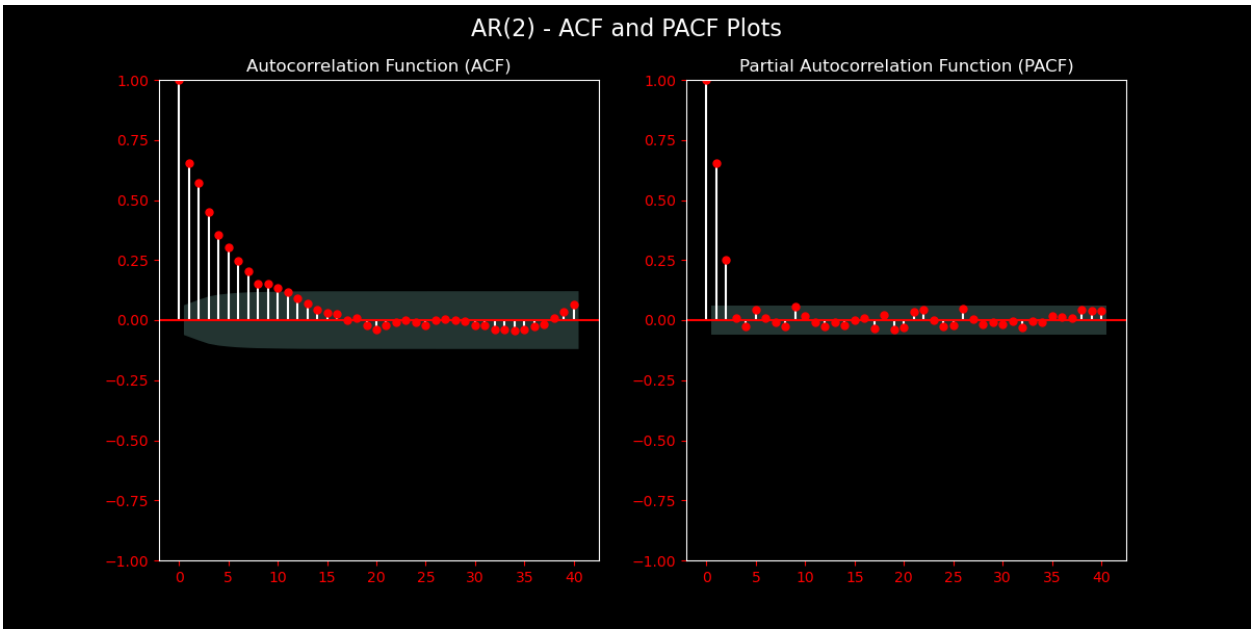
Due to these conclusions on the AR(2) and MA(1) processes, I label this as a ARMA(2, 1) process.

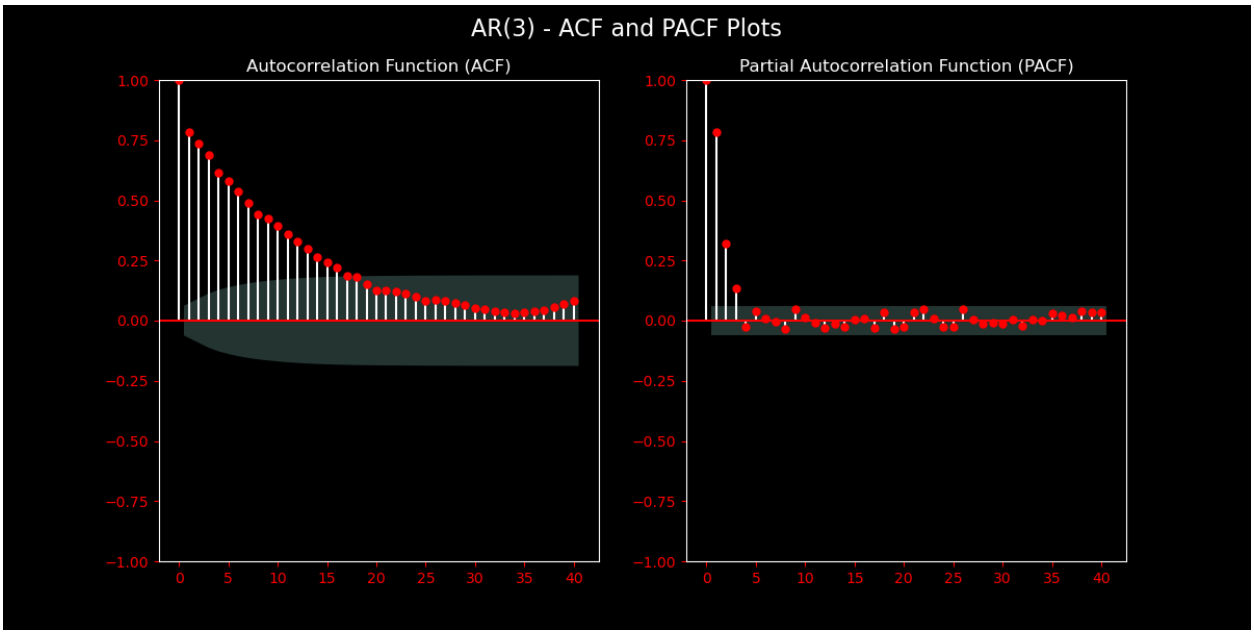


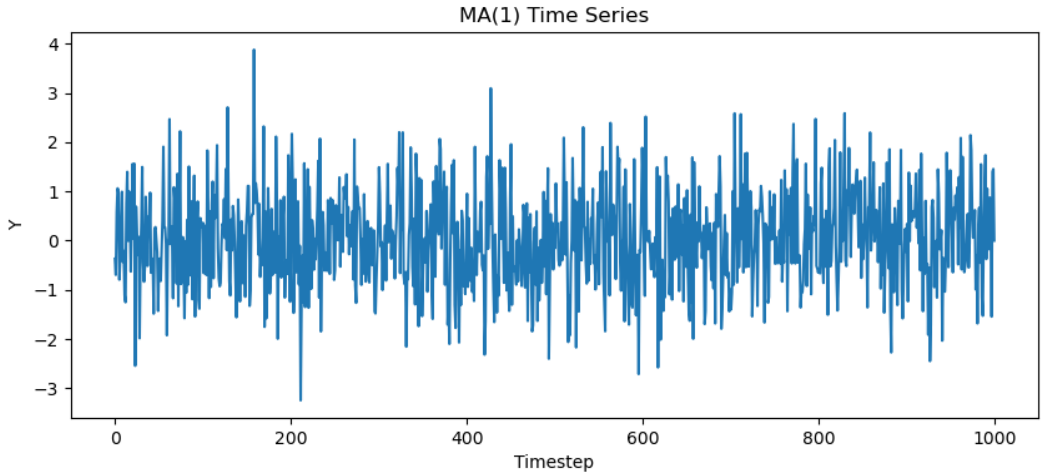


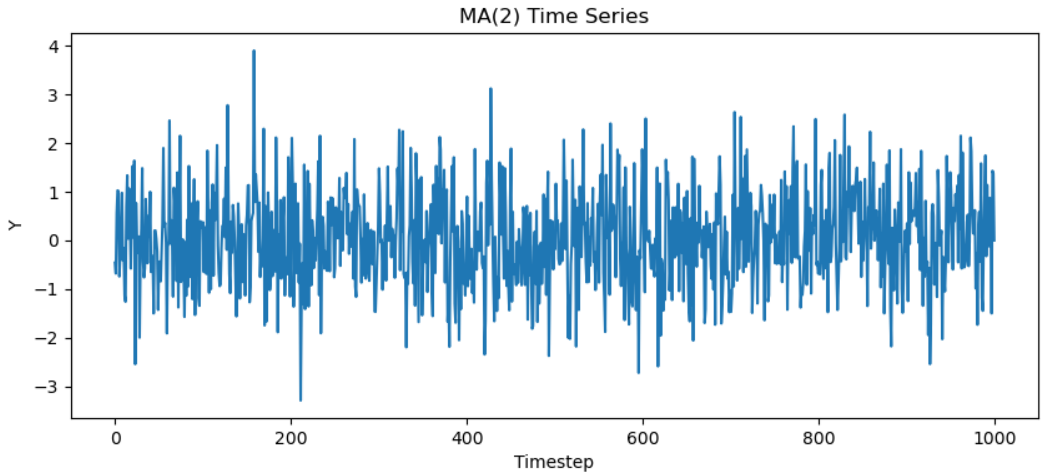

AR(3) Time Series
Y
Timestep
4
2
0
−2
−4
0
200
400
600
800
1000

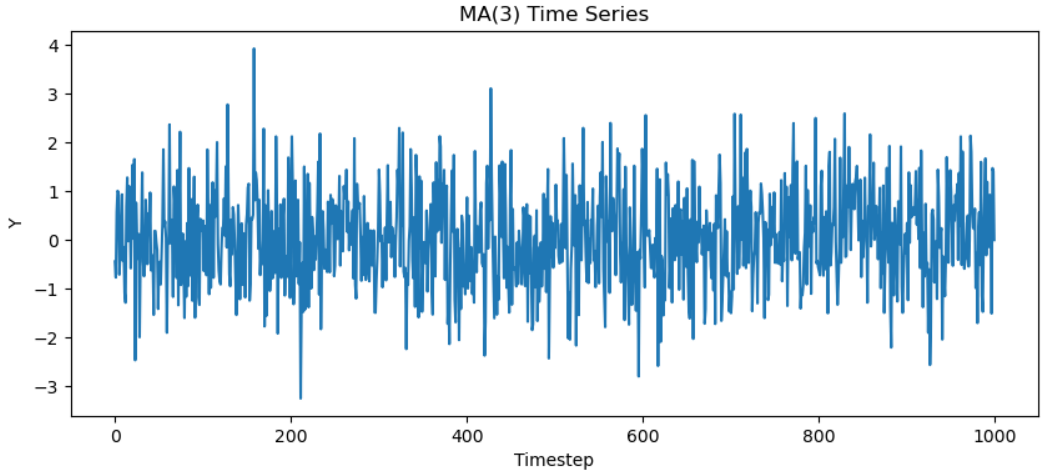

AR(1) - ACF and PACF Plots
Autocorrelation Function (ACF)
Partial Autocorrelation Function (PACF)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
30
35
40

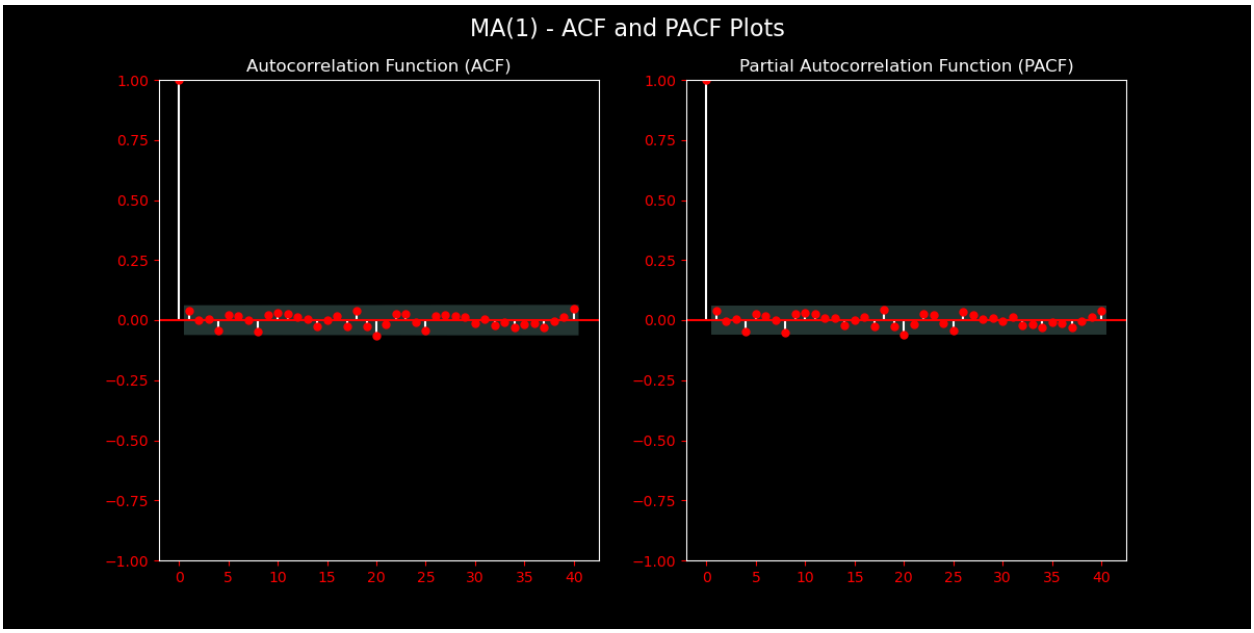

AR(2) - ACF and PACF Plots
Autocorrelation Function (ACF)
Partial Autocorrelation Function (PACF)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
30
35
40

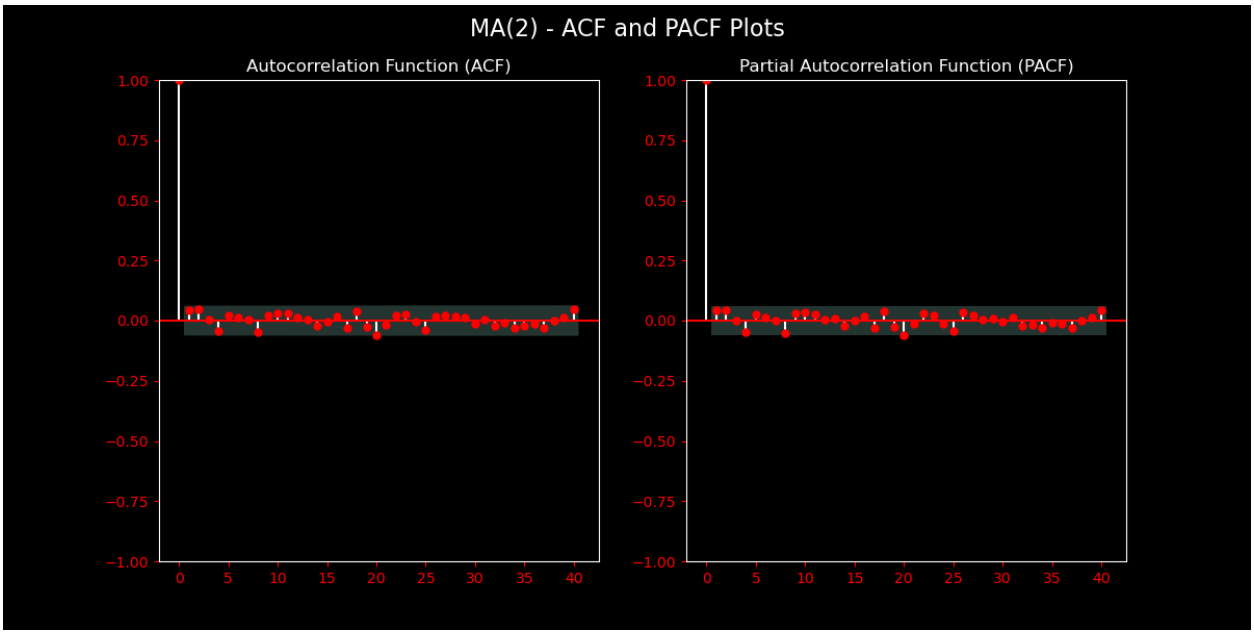

AR(3) - ACF and PACF Plots
Autocorrelation Function (ACF)
Partial Autocorrelation Function (PACF)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
30
35
40

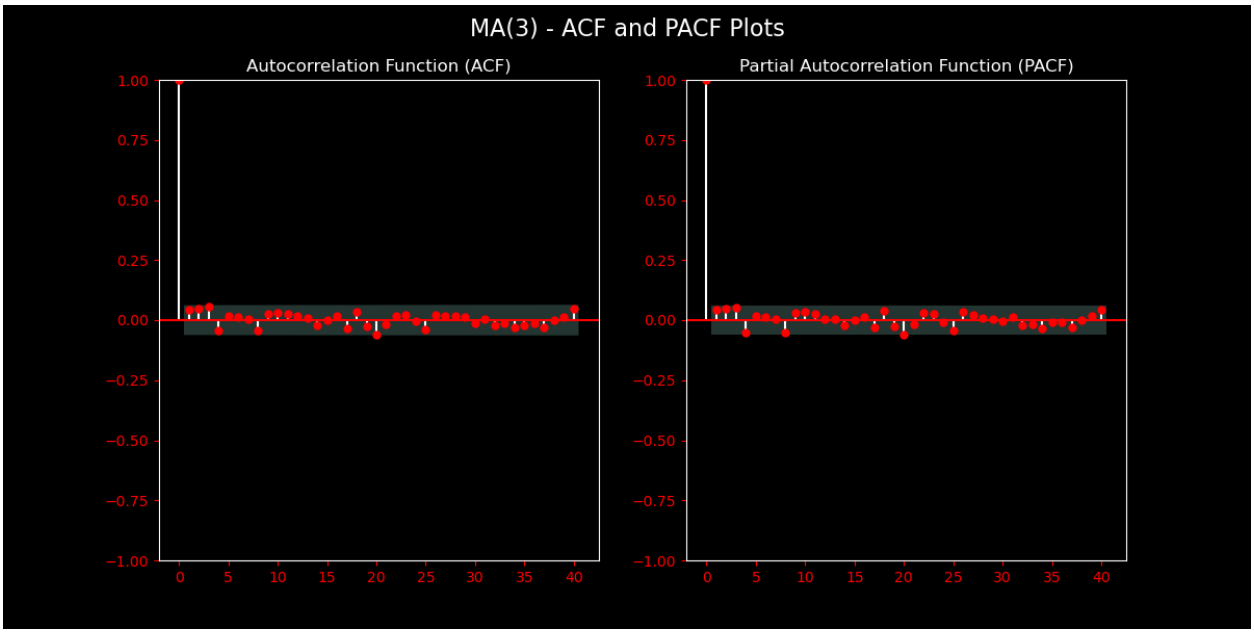

MA(1) Time Series
Y
Timestep
4
3
2
1
0
−1
−2
−3
0
200
400
600
800
1000


MA(2) Time Series
Y
Timestep
4
3
2
1
0
−1
−2
−3
0
200
400
600
800
1000


MA(3) Time Series
Y
Timestep
4
3
2
1
0
−1
−2
−3
0
200
400
600
800
1000


MA(1) - ACF and PACF Plots
Autocorrelation Function (ACF)
Partial Autocorrelation Function (PACF)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
30
35
40


MA(2) - ACF and PACF Plots
Autocorrelation Function (ACF)
Partial Autocorrelation Function (PACF)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
30
35
40


MA(3) - ACF and PACF Plots
Autocorrelation Function (ACF)
Partial Autocorrelation Function (PACF)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
30
35
40