Codes to all the problems are included in the python (ipynb) file in github

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.

Following the steps mentioned in the notes from week 01, I calculated the kurtosis and then created a sample of 100 kurtosis and calculated its mean and standard deviaton. Then I calculated the t-statistic for the distribution and used that to find the p-value using the CDF function. If the p-value was lower than the threshold value of 0.05, then we reject the hypothesis and conclude that the kurtosis function is unbiased. However, we noticed that the p-value is greater than the threshold for every single instance confirming that the statistical package function is biased.

I also calculated the t-statistic using scipy's built in t-test function and the p-value does not significantly change (changes in the 3rd decimal point).

Here is a sample code snippet. Details are provided in the code file.

```
        print(f"Failed to reject the hypothesis and kurtosis function is biased (p-value = {p_value})")
```

```
Failed to reject the hypothesis and kurtosis function is biased (p-value = 0.16740819247694438)
```

Trying the alternative method using t-test function from the package

```
from scipy.stats import ttest_1samp

# Step 4: Calculate the mean kurtosis k
mean_sampled_kurtosis = np.mean(sampled_kurtosis)

# Perform one-sample t-test
t_statistic, p_value = ttest_1samp(sampled_kurtosis, popmean=0)

# Print the results
print(f"t-statistic: {t_statistic}")
print(f"P-value: {p_value}")
```

```
t-statistic: -1.3736595741584392
P-value: 0.17264998026793738
```

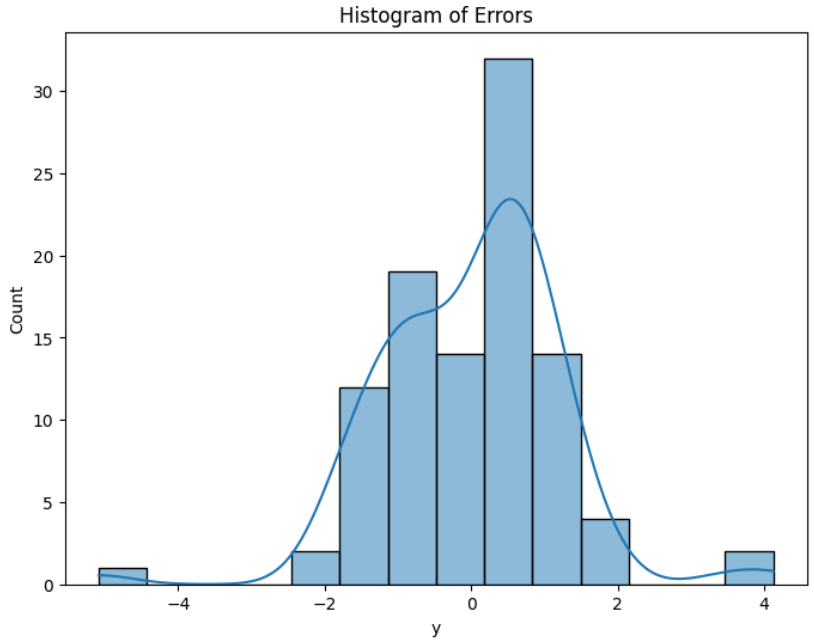
Problem 2

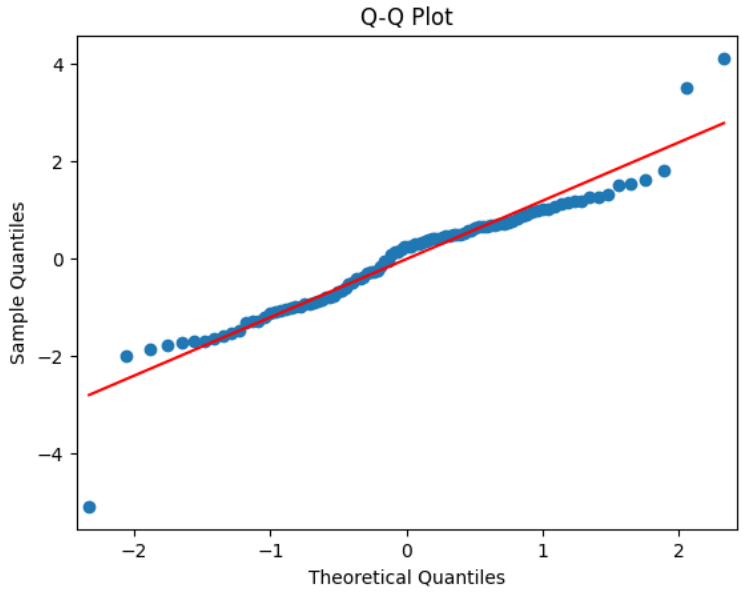
Fit the data in problem2.csv using OLS and calculate the error vector. Look at its distribution. How well does it fit the assumption of normally distributed errors?

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 regard to expected values in this case?

The OLS fit provided the B0(Beta 0) estimate of 0.12 and B1(Beta 1) estimate of 0.61. The plots below show the histogram and Q-Q plots for the error vectors. The plots depict how the error vector is normally distributed. The histogram here does have somewhat bell shape however it is not normally distributed in the center and is skewed to the right. Similarly, the Q-Q plot above checks for the assumption of normality if the points closely follow the diagonal line. The points here somewhat follow the diagonal line confirming the errors are not normally distributed.






```
                            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:                Sun, 10 Sep 2023   Prob (F-statistic):           4.34e-06
Time:                        03:21:56   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
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

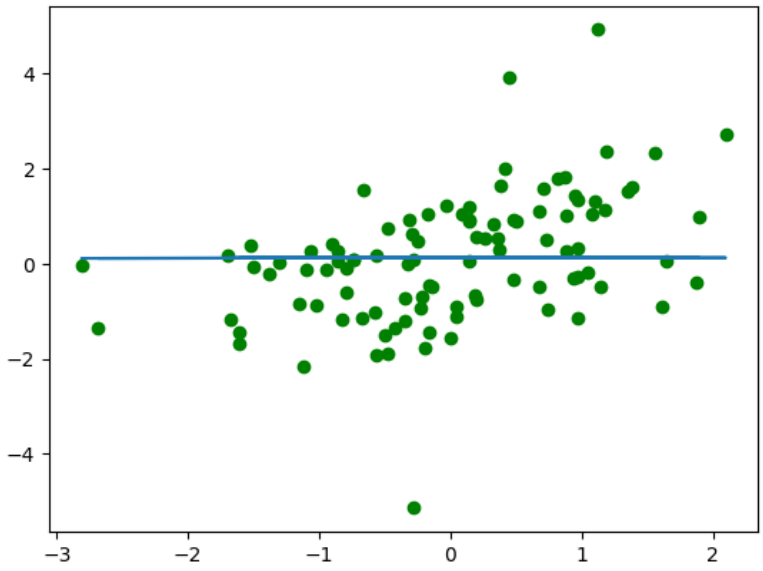
Fit the data using MLE given the assumption of normality. Then fit the MLE using the assumption of a T distribution of the errors. Which is the best fit?

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 regard to expected values in this case?

The data has been fit using MLE on the assumption of normality and T distributon of the errors. Below the first two images show the results of the MLE on the assumption of normality and the last one shows the assumption based on T-distributon of the errors. The MLE estimate was performed using L-BFGS-B, which is an exenson of the BFGS algorithm, which is a quasi-Newton method for unconstrained optimization. The L-BFGS-B handles bounded parameter spaces. We noticed that MLE based on normality converges to the following values for intercept, slope and standard deviation. However, the MLE based on T-distribution fail to converge. It remains at the initial guess value. For both the fits, I used the same initial guess values of 1,1,1 for intercept, slope and standard deviation. The T distribution consists of another parameter, degrees of freedom, which was chosen to be 3 at random. The fitted parameters for the MLE using normality are given as follows: beta0(intercept) = 0.022; beta1(slope) = 88.63, and sigma (standard deviation) = 1.34. Since the other MLE failed to converge, there was no fitted parameters available.

The intercept does not directly tell us about the normality assumption. It is important to note that the normality assumption relates to the distribution of the errors and not the intercept. Similarly, the beta1 does not provide anything about normality assumption. However, the standard deviation of errors, which is 1.34, represents the deviations of the observed Y-values from the fitted values on the regression line. If the normality assumption holds, the distributions of these errors should be approximately normal with mean zero. A small value of sigma suggests that the model is providing a good fit to the data.

```
MLE Estimate for Intercept (beta0): 0.02246111787974192
MLE Estimate for Slope (beta1): 88.63005235734333
MLE Estimate for Standard Deviation (sigma): 1.3353755226441624
```




```
...      message: CONVERGENCE: NORM_OF_PROJECTED_GRADIENT_<=_PGTOL
         success: True
          status: 0
             fun: -0.0
               x: [ 1.000e+00  1.000e+00  1.000e+00  4.000e+00]
             nit: 0
             jac: [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00]
            nfev: 5
            njev: 1
        hess_inv: <4x4 LbfgsInvHessProduct with dtype=float64>
    MLE Estimate for Intercept (beta0): 1.0
    MLE Estimate for Slope (beta1): 1.0
    MLE Estimate for Standard Deviation (sigma): 1.0
    MLE Estimate for Degrees of Freedom (nu): 4.0
```

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?

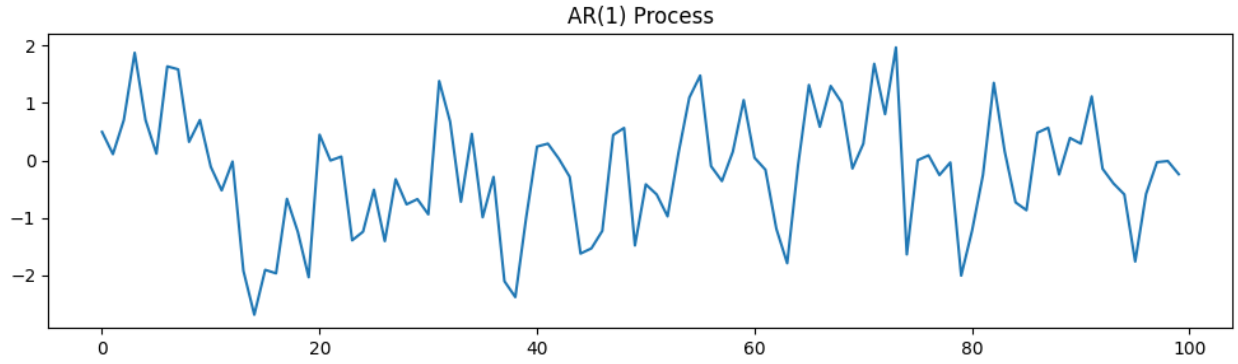
AR(p) Process, where p denotes 1 through 3

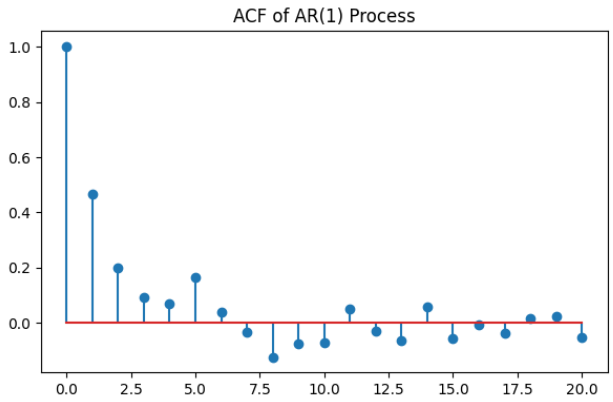
- ACF is exponentially decreasing after a few initial lags
- PACF shows a sharp cutoff after p lags.

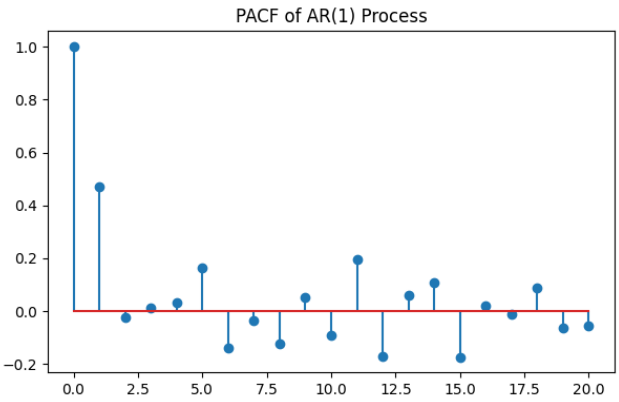
MA(q) Process

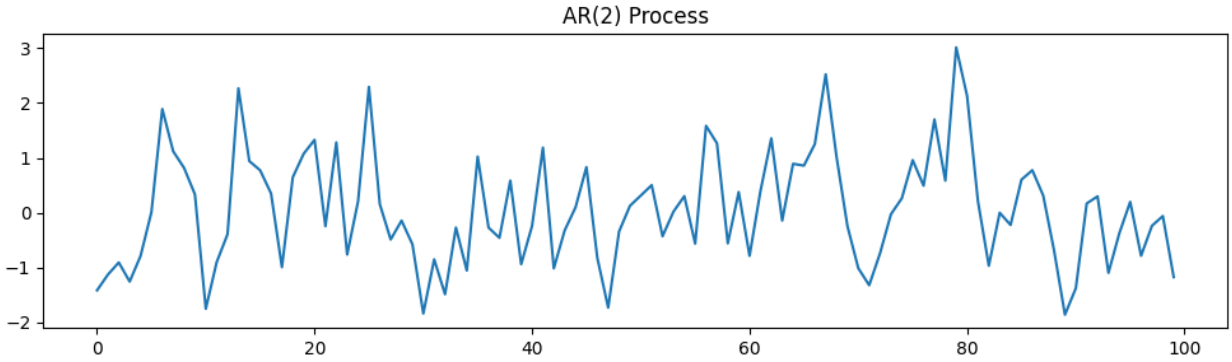
- ACF shows a sharp cutoff after lag q
- PACF is exponentially decreasing or cuts off after a few lags

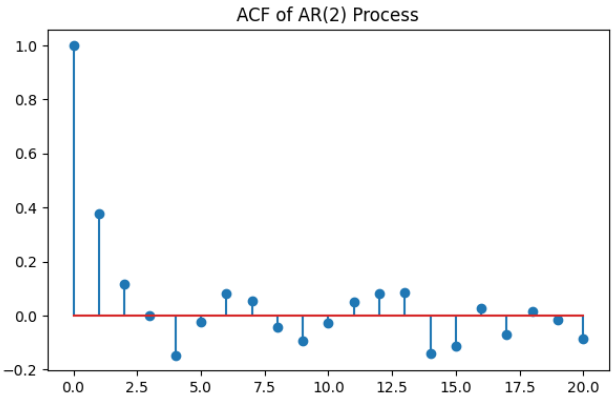
All the graphs generated in VS code are attached below. They consists ACF and PACF graphs for both AR(1) through AR(3) and MA(1) through MA(3)

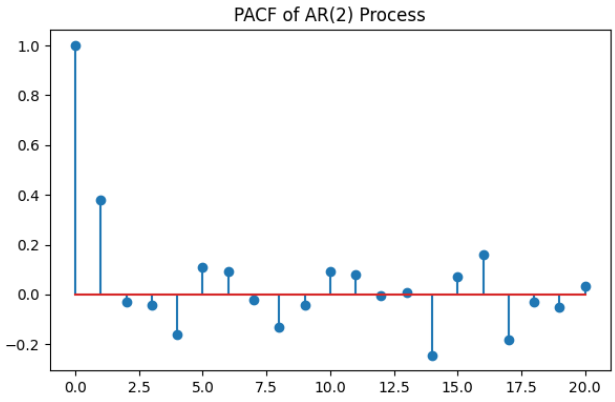


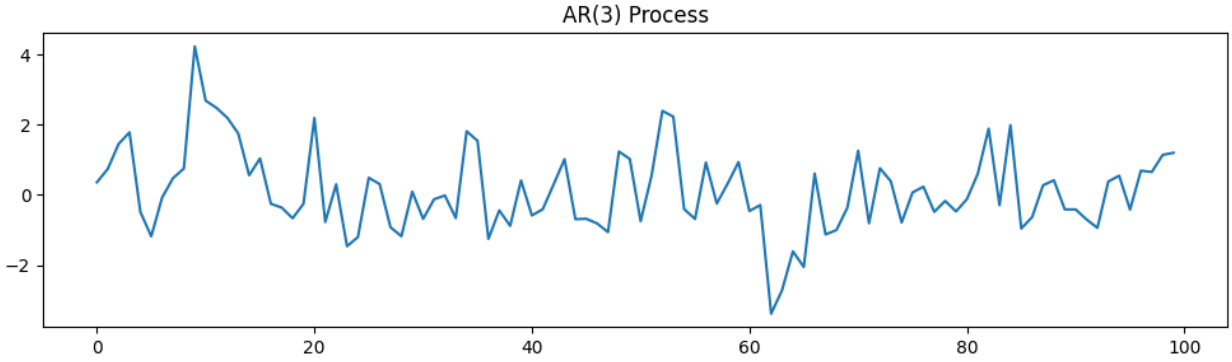


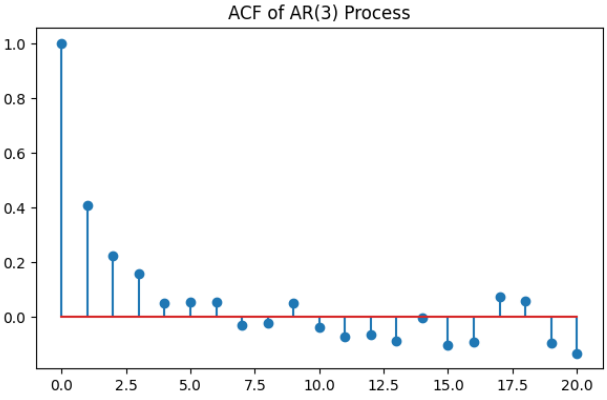


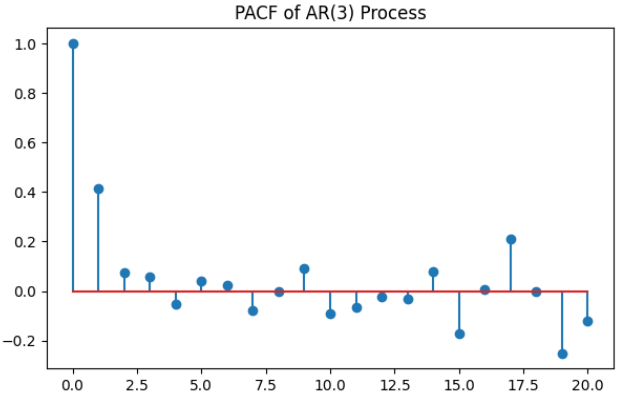

AR(2) Process
3
2
1
0
−1
−2
0
20
40
60
80
100

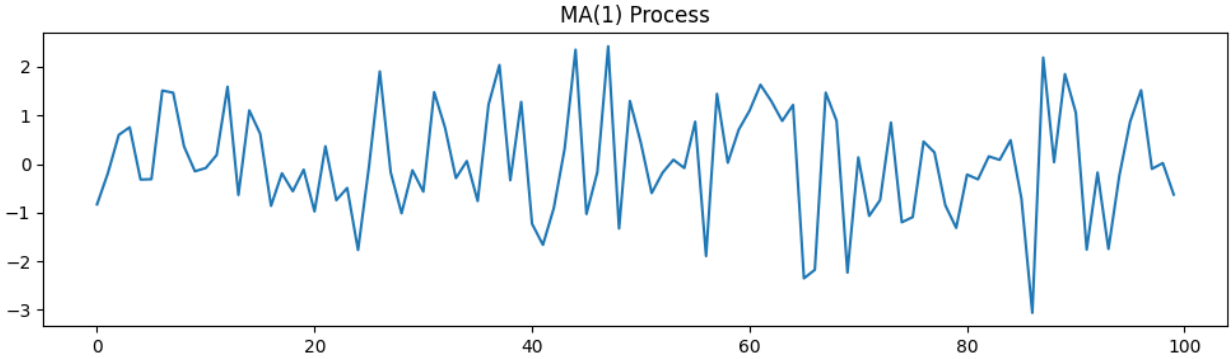

ACF of AR(2) Process
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

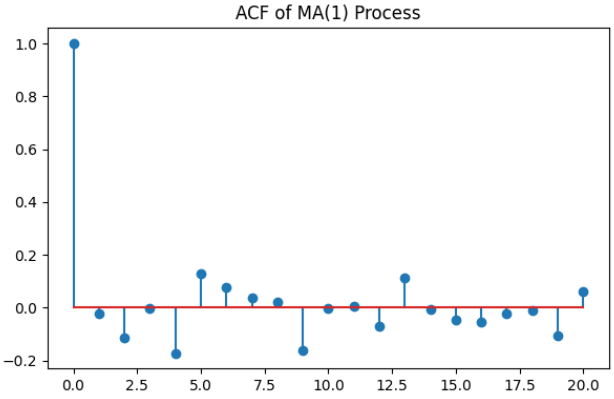

PACF of AR(2) Process
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

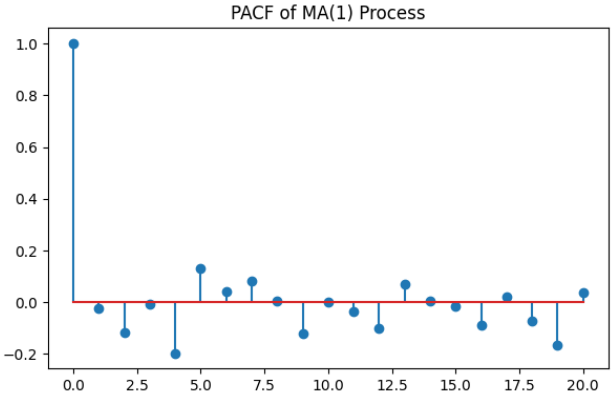

AR(3) Process
4
2
0
−2
0
20
40
60
80
100

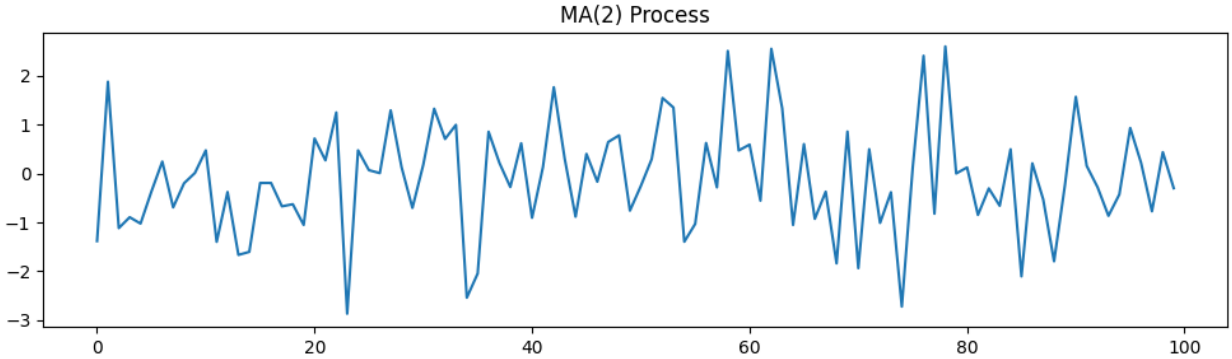

ACF of AR(3) Process

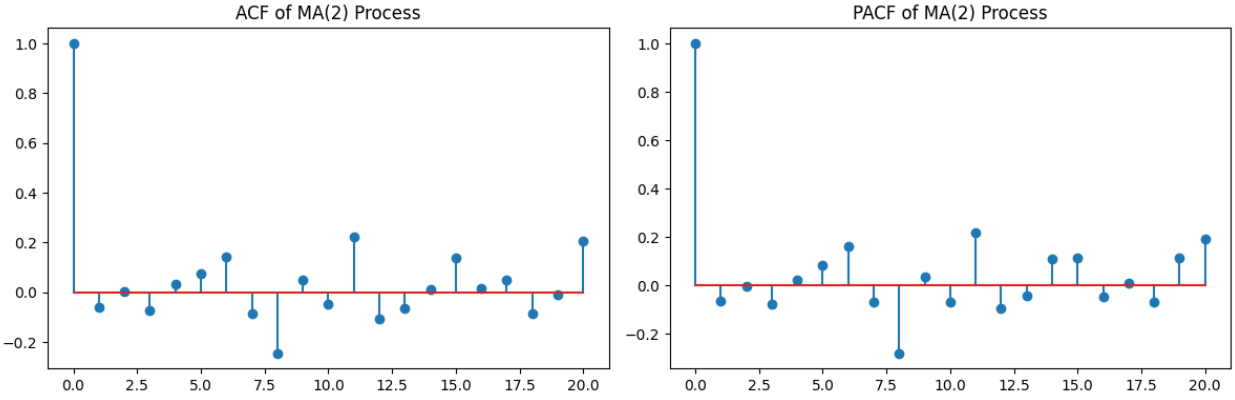

PACF of AR(3) Process

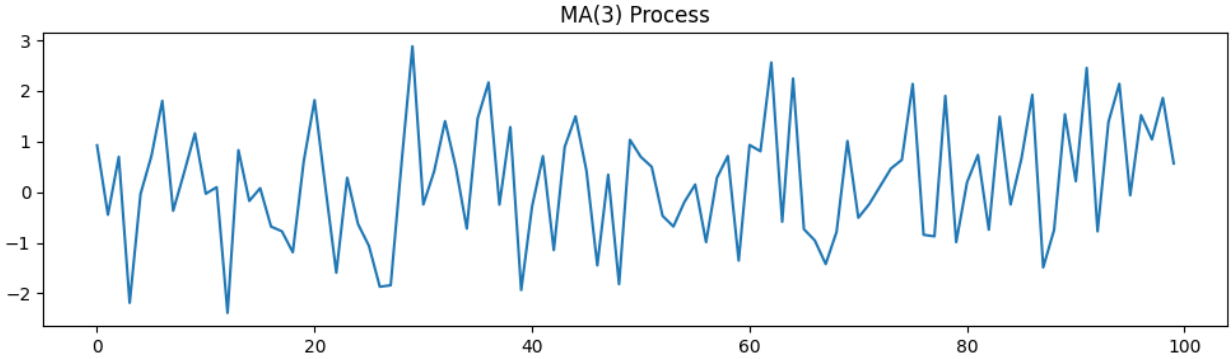

MA(1) Process

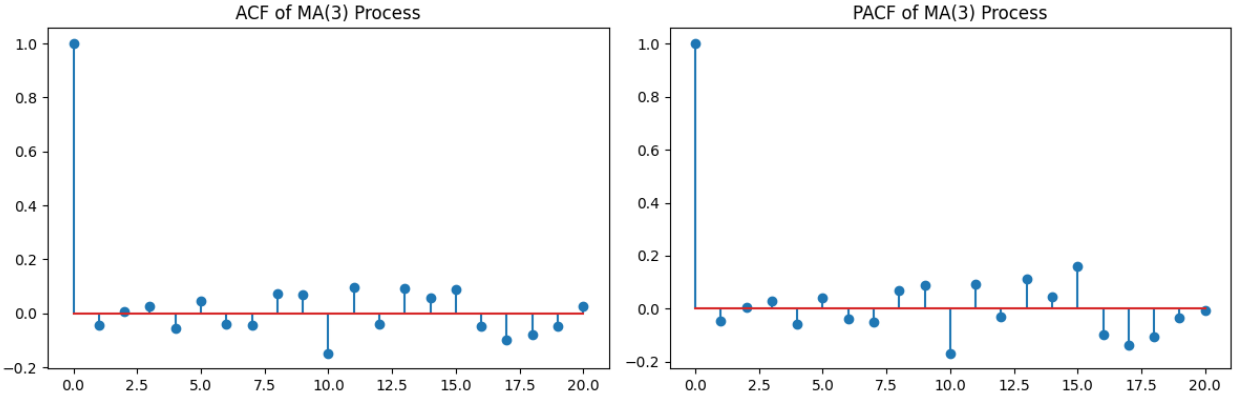

ACF of MA(1) Process


PACF of MA(1) Process


MA(2) Process


ACF of MA(2) Process
PACF of MA(2) Process


MA(3) Process


ACF of MA(3) Process
PACF of MA(3) Process