Name: Xiaokuan Zhao
ID: xz358

Problem1:

The result is the skewness and kurtosis functions provided in the scipy.stats is unbiased.

Here is how I approach this.

I want to test if the skewness and kurtosis functions in statistical packages are biased or not, so I created a list of random data with the length as the input in a function.

Inside the function, I first use the skewness and kurtosis functions provided in the scipy.stats to calculate the number by computer, then I use the definition of skewness and kurtosis to calculate by hand. After that, I calculate the difference. Note the kurtosis function in scipy.stats uses a different normalization than the one in excel, so it returns excess kurtosis, that is the kurtosis of the data minus 3, that is why I plus my result with 3.

I use 100, 500, and 1000 as the length of data to see the difference and this is the output result:

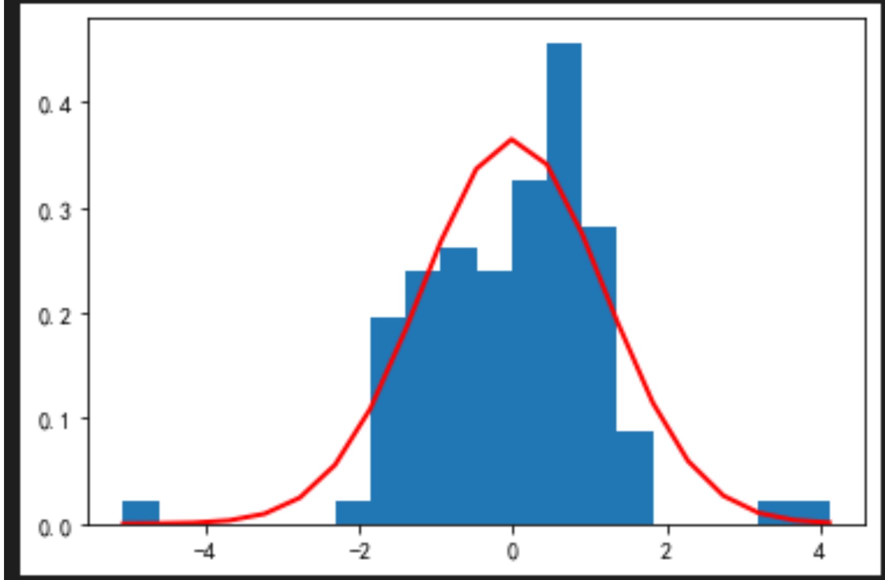
```
when n= 100  the Skewness by function: -0.18228321437425338
when n= 100  the Kurtosis by function: 2.7345641275230625
when n= 100  the Skewness by hand: -0.18228321437425343
when n= 100  the Skewness by hand: 2.734564127523062
Skewness difference: 5.551115123125783e-17
Kurtosis difference: 4.440892098500626e-16
when n= 500  the Skewness by function: -0.19819337483554447
when n= 500  the Kurtosis by function: 2.882788112081306
when n= 500  the Skewness by hand: -0.19819337483554453
when n= 500  the Skewness by hand: 2.8827881120813066
Skewness difference: 5.551115123125783e-17
Kurtosis difference: -4.440892098500626e-16
when n= 1000  the Skewness by function: -0.009009133223947984
when n= 1000  the Kurtosis by function: 3.1412629458999697
when n= 1000  the Skewness by hand: -0.009009133223947984
when n= 1000  the Skewness by hand: 3.141262945899971
Skewness difference: 0.0
Kurtosis difference: -1.3322676295501878e-15
```

Because the difference between the number computed by function and hand is very close, it is safe to say that the skewness and kurtosis functions provided in the scipy.stats is unbiased.

Problem2

I used the OLS in statsmodels to apply the OLS Regression on the data and calculated the error as residual in the code. Then I used MOM Estimation to calculate its mean and standard error. With those, I can plot a normal distribution using both parameters and draw the graph as follows.

```
According to MOM estimation
The mean of the residual is: -2.1094237467877975e-17
The standard deviation of the residual is: 1.1983941277418964
```



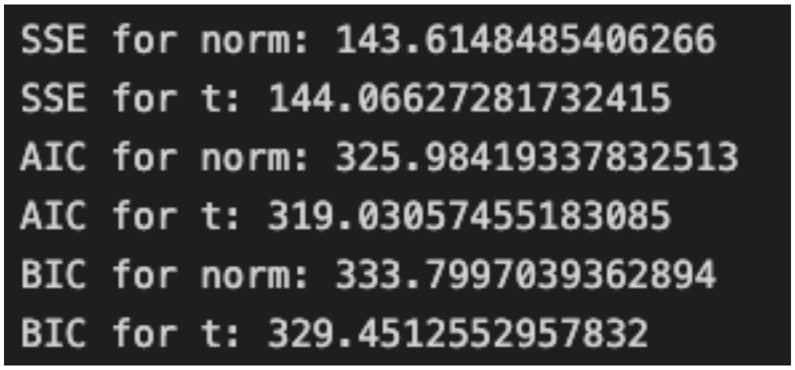
In the graph, the density of error term is represented by the blue hist. The red line is the normal distribution. So, from the graph, we can easily tell the error terms does not *seem* like normal distribution. Then I use the Shapiro-Wilk test to test my hypothesis. The null hypothesis is: Data provide is normal distributed. The P value is 0.000154, so we can reject the null hypothesis so the error term is not normal distribution.

After fitting the model with the assumption that error is normal or T distribution, this is what I get:

```
the k in OLS =  0.6052
the b in OLS =  0.1198
the k in MLE with normal distribution =  0.6052048882824387
the b in MLE with normal distribution =  0.11983619162658825
the k in MLE with T distribution =  0.5589304973270935
the b in MLE with T distribution =  0.06998057376341564
```

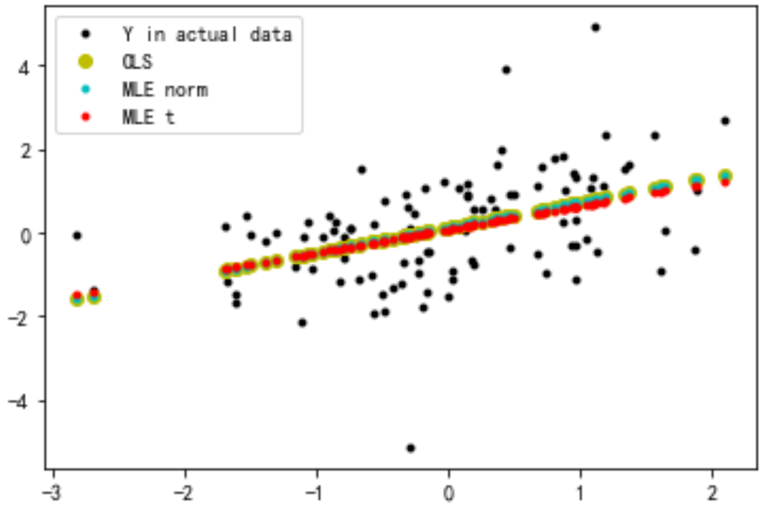
We can see that with different assumption, the coefficient is different.

In order to compare which is a better fit, I use SSE, AIC and BIC to test and here is the result:



```
SSE for norm: 143.6148485406266
SSE for t: 144.06627281732415
AIC for norm: 325.98419337832513
AIC for t: 319.03057455183085
BIC for norm: 333.7997039362894
BIC for t: 329.4512552957832
```

We can see the SSE for normal distribution assumption is lower however the AIC and BIC is lower for T distribution by much so I think T distribution is better here.

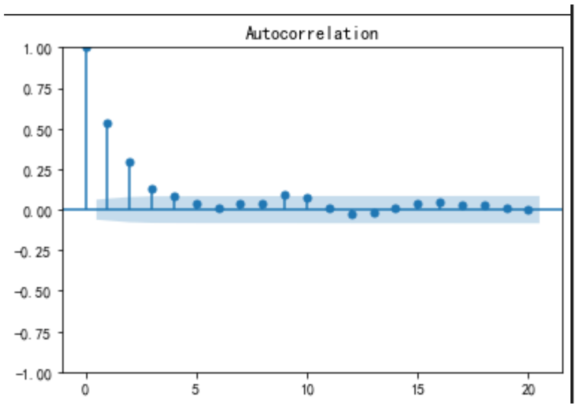


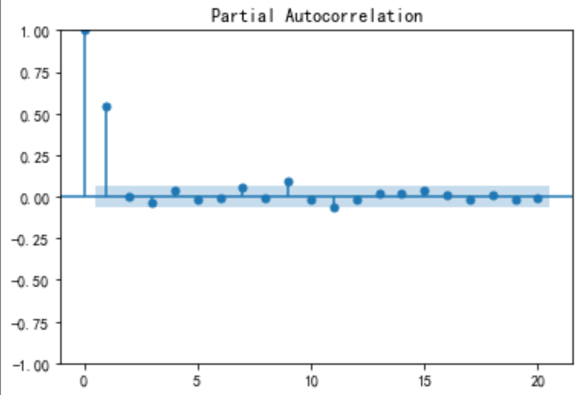
From this graph, we know that when the assumption of the error term is different, the coefficient and constant of the regression will change accordingly, make it different than the OLS regression.

Problem 3

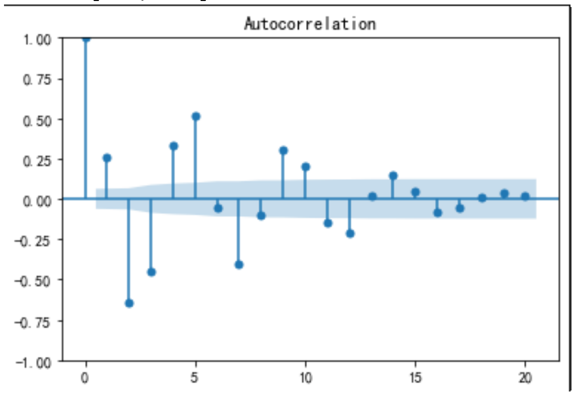
Problem 3 asks to simulation AR(1) to AR(3) and MA(1) to MA(3) and draw their ACF and PACF to identify the type and order of each process. I will report the images of ACF and PACF below and analyze it afterwards. Of course, when generating all the processes, I must make sure all processes are stationary or the analysis will not be effective.

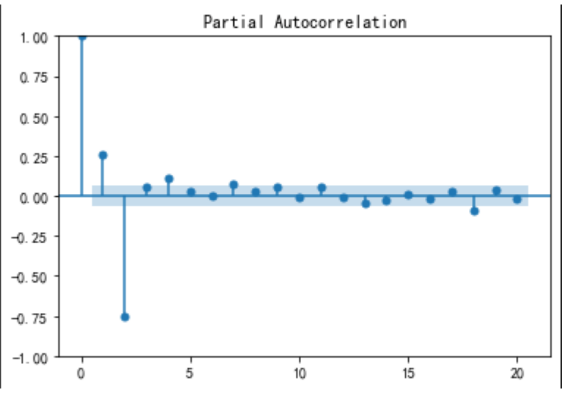
AR(1)
coef = 0.5



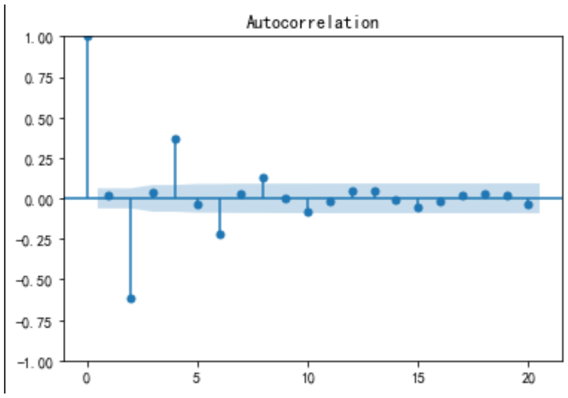


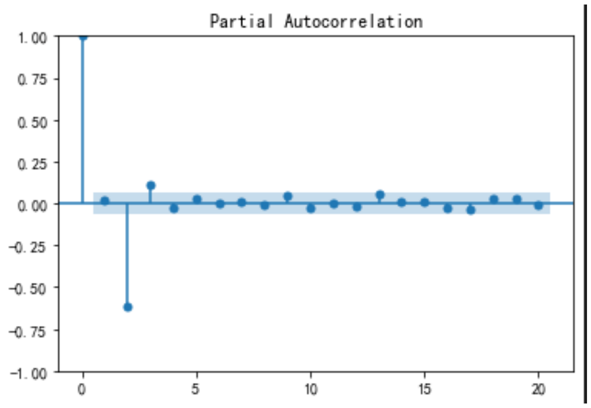
AR(2)
coef = [0.4,-0.5]



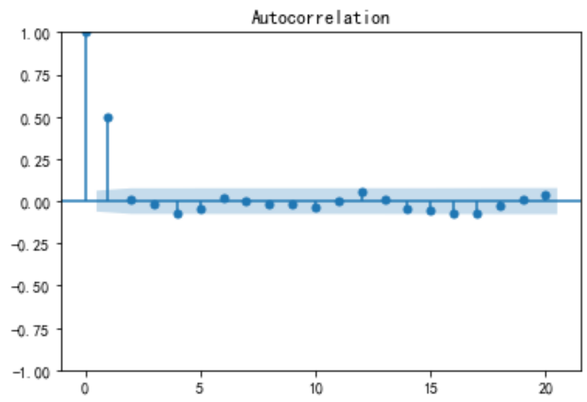


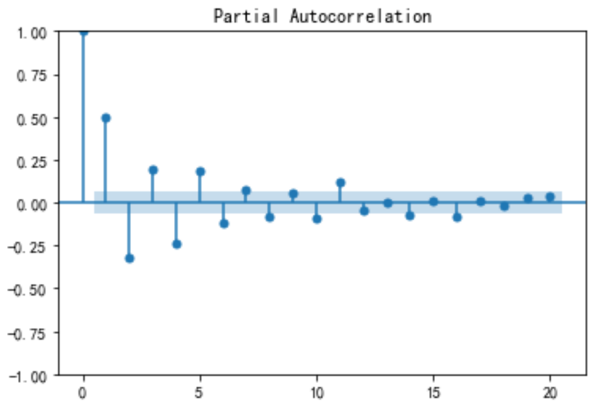
AR(3)
coef = [0.1,-0.6,0.1]



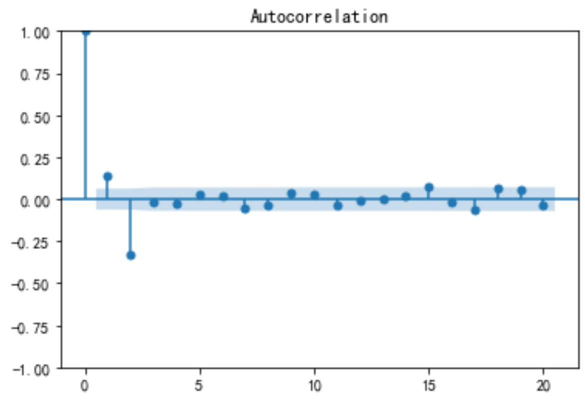


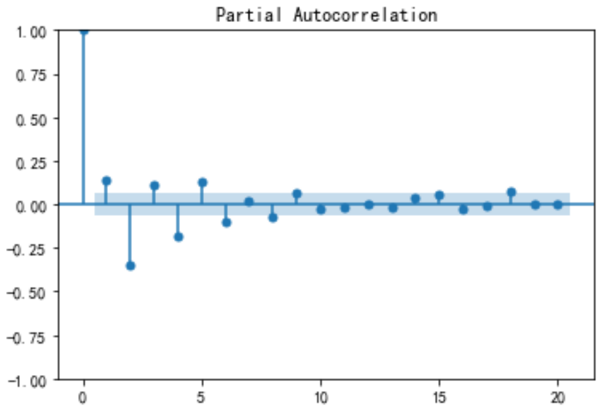
MA(1)
theta = 0.9



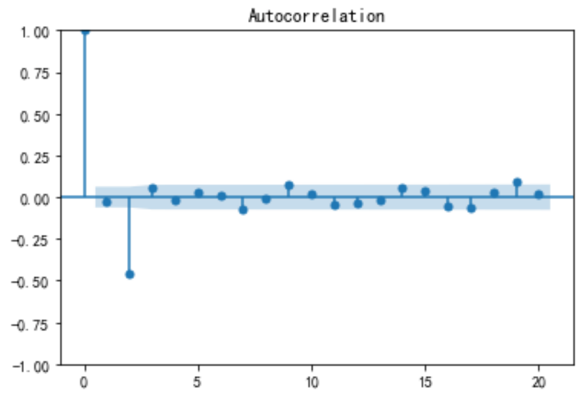


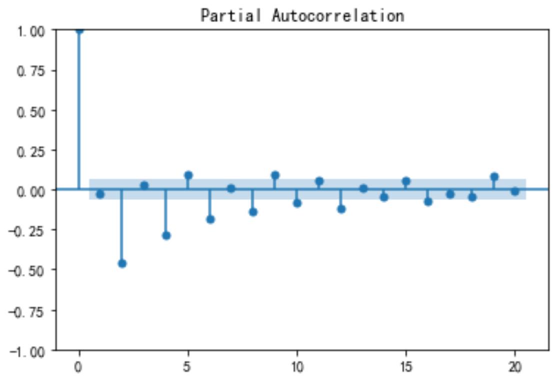
MA(2)
theta1 = 0.3 theta2 = -0.4





MA(3)
theta1 = 0.1 theta2 = -0.7 theta3 = 0.1





From the graphs, we know that there are differences between MA and AR processes.

In an MA process, the ACF will have significant values at the first lag and then tail off very quickly. The PACF will have significant values for all lags relative to the MA model, outside the confidence interval. We can determine the order of an MA process by finding the last significant tall lag in the ACF graph. But it is not so accurate and the specific order should be determined with more tools, but this method can narrow it down.

In an AR process, the PACF will have significant values at the first lag and then tail off very quickly. The ACF will have significant values for all lags relative to the MA model, outside the confidence interval. We can determine the order of an AR process by finding the last significant tall lag in the PACF graph. But it is not so accurate and the specific order should be determined with more tools, but this method can narrow it down.

To sum up, an MA process has an ACF graph that tailing off very quickly and an AR process has a PACF graph that tailing off very quickly. This can be used to distinguish from each other. The order can be determined by looking at its last high lag.