Response for Project Week02

Course Code: Fintech545

Course Title: Quantitative Risk Management

Student Name: Wanglin (Steve) Cai

Student Net ID: WC191

Problem 1

For this problem, I select Python as the programming language. Therefore, I examined the statistical packages in Python for this problem.

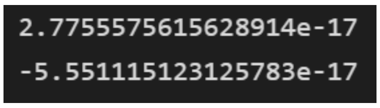
The package that I examined is **scipy.stats**. The functions related to skewness and kurtosis in this package is called **skew()** and **kurtosis()**. To compare whether these two functions are biased. I built two biased functions (shown below, where s is the standard deviation of the sample, X is the data sample and N is the number of samples), since they are both divided by the sample size, which is not the case for unbiased functions.

$$Skewness = \frac{\sum_{i=1}^{N}(x_i - \overline{x})^3/N}{s^3}$$

$$Kurtosis = \frac{\sum_{i=1}^{N}(x_i - \overline{x})^4/N}{s^4}$$

For the data part, I generated two datasets, both of them are from standard normal distribution. The main difference is that the first dataset has 100 data point and the second has 1000 datapoint. In this way, I think the result can be more robust.

Then, I compared the results from my owned function and the results from the package built-in function by taking the difference of them. I found that the difference is extremely close to 0, which means that we can pretty sure that the built-in functions in the **scipy.stats** package are biased. The reason why the differences are not equals to exactly 0 is probably because the computer's rounding problem in the steps of calculations.



```
2.7755575615628914e-17
-5.551115123125783e-17
```

Fig 1. The Difference of Skewness Between Built-in Functions in Scipy.stats and the Biased Functions that I Built for Dataset1 and Dataset2

After searching for some documents, I became even more sure that the built-in functions are biased. Because these functions have a parameter called bias, and by default this parameter is set to 'True', which means that the skewness and kurtosis functions are biased by default.

In conclusion, the skewness and kurtosis function in this Python statistical packages are biased.

Problem 2

For the first part of this problem, it basically asked us to first fit the data using OLS, and then examine the assumption of normally distributed errors.

For the second part, it asked us to fit the data using a different technique (i.e., MLE) with different assumptions of errors (normal distribution/T distribution), then make a comparison to see which one fits better.

For the last part, it asked us to compare the fitted parameters and draw some conclusions.

In the first part, after fitting the data with OLS and get the error vectors, the distribution of error vectors is shown below.

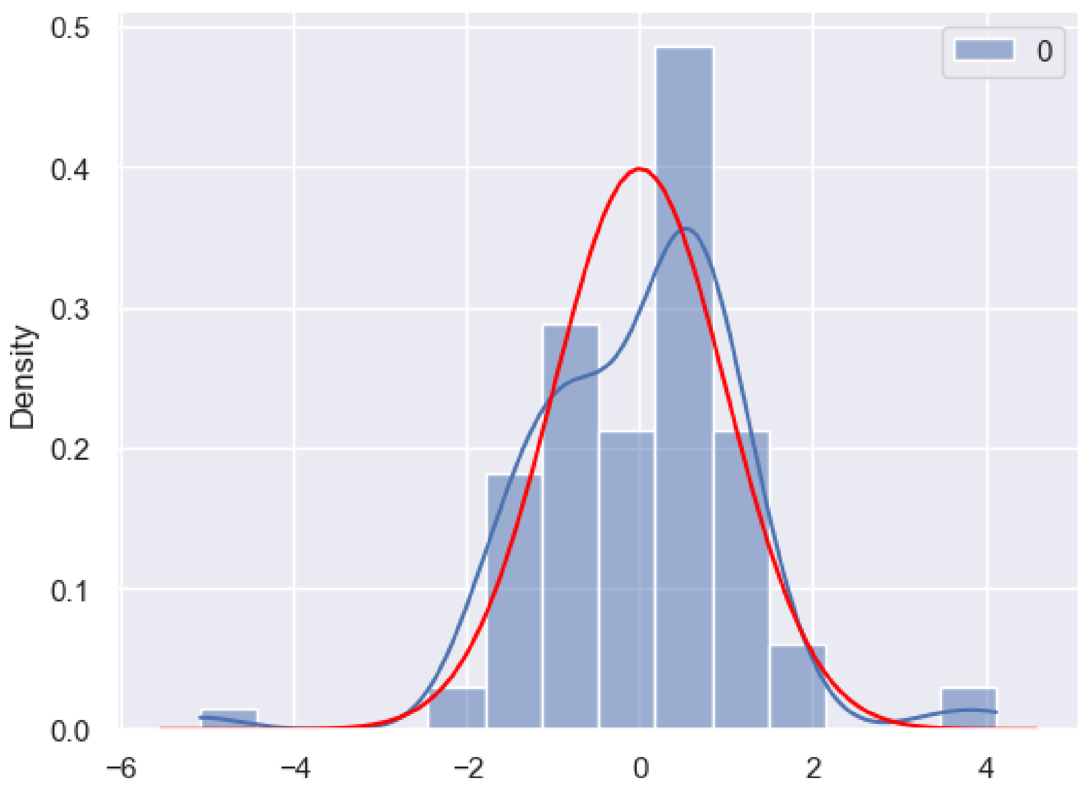


Fig 2. The Distribution of Error Vectors using OLS Compared to The Standard Normal Distribution

We can easily see that the skewness of the distribution of error vectors is negative, in this case the skewness is roughly -0.3, which means that the distribution has a fatter tail on the right side. Besides, compared to the normal distribution, the kurtosis of this distribution is positive, in this case greater than 3, which also reveals the fatter tail compared to the normal distribution.

Therefore, from the distribution, we can draw a conclusion that the assumption of normally distributed errors is untenable. From the plot, the distribution of error vectors reveals that the underlying distribution might be a distribution with a fatter tail. For example, student t-distribution could be a more reasonable guess in this case.

For the second part, with another techniques MLE and given the assumption of normality, we get the result of beta hat and the error vectors. In this case, the beta hat and the error vectors are exactly the same as the previous case.

For the assumption of t-distribution, since we do not know the degree of freedom in the t-distribution of the error term, I regarded the degree of freedom as a special kind of hyper parameter. In other words, we can freely choose the degree of freedom in the t-distribution. In my case, I choose degree of freedom to be 2 and 10. In the case that degree of freedom = 2, we can see the kurtosis from distribution of error terms is very close to the kurtosis of the t-distribution with degree of freedom = 2. In the case that the degree of freedom = 10, we can see the kurtosis from t-distribution with degree of freedom = 10 is not quite equal to the kurtosis from the distribution of error terms.

Simply from the graphs below, I think the MLE with assumption of t-distribution with degree of freedom = 2 is the best fit overall.

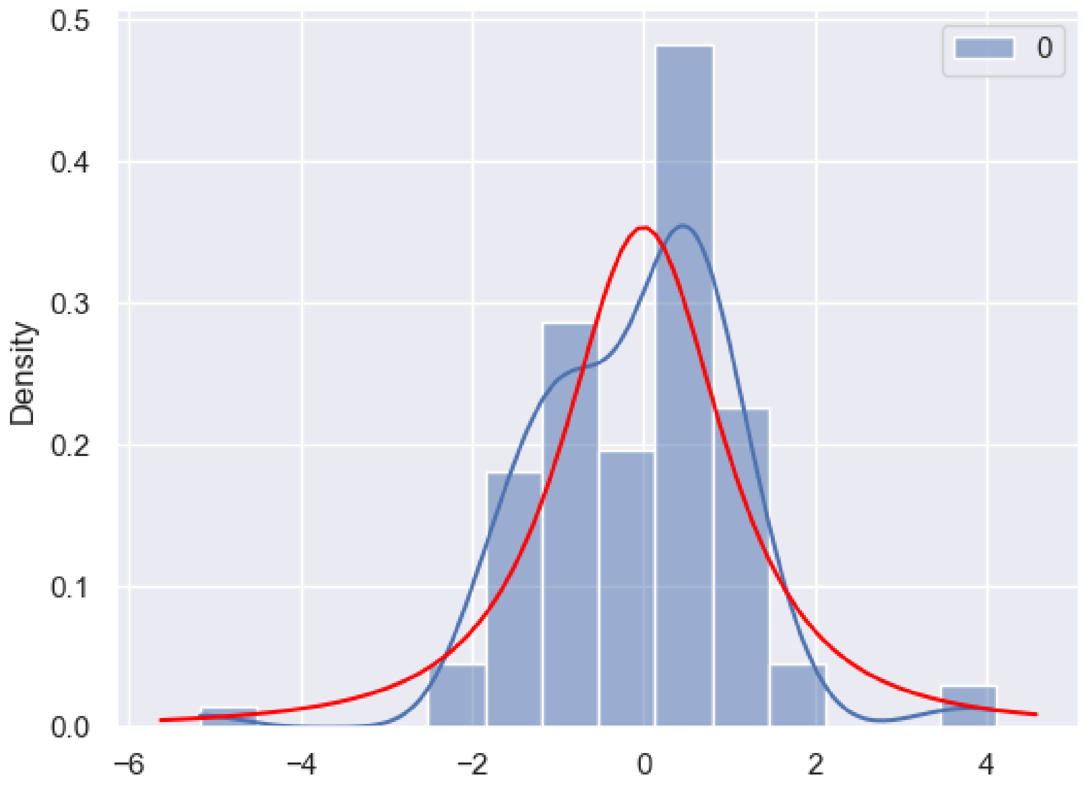


Fig 3. The Distribution of Error Vectors using MLE Compared to T-Distribution with Degree of Freedom = 2

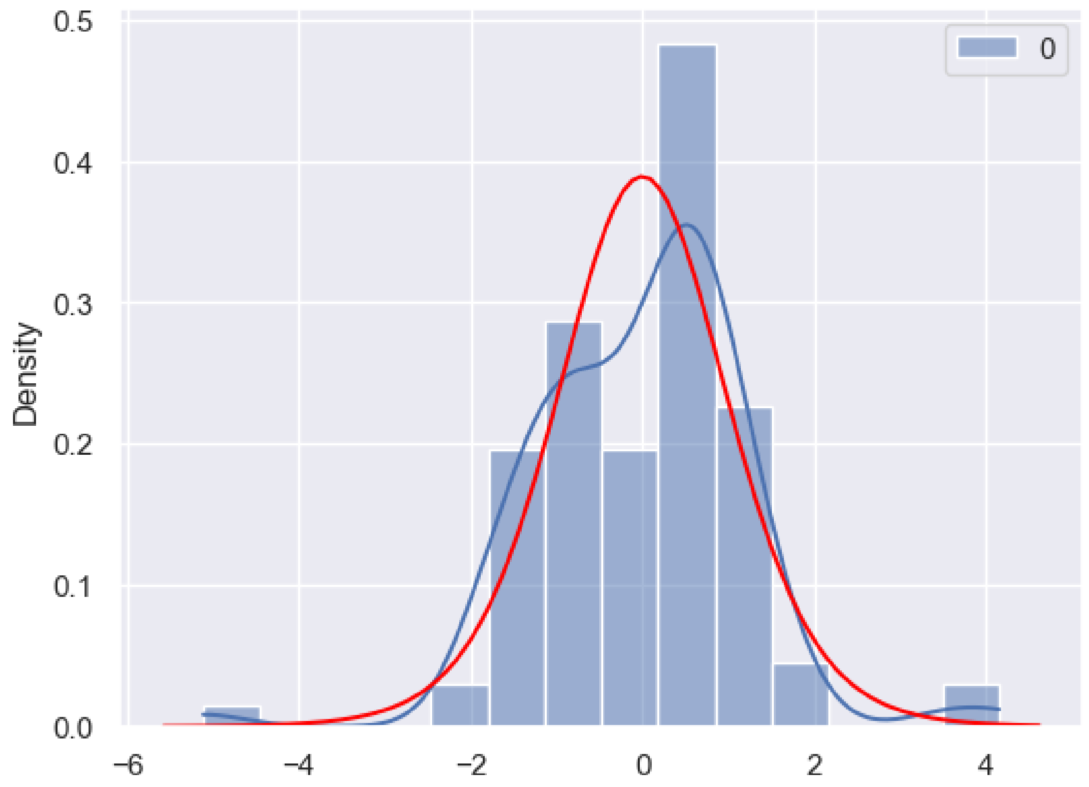


Fig 4. The Distribution of Error Vectors using MLE Compared to T-Distribution with Degree of Freedom = 10

However, we can also look at the criteria of goodness of fit such as r squared. To our surprise, simply from the goodness of fit criteria perspective, we found that the normality assumption is better than the t-distribution assumption.

METHOD	$\boldsymbol{R^2}$
OLS, normal distribution	0.1946
MLE, normal distribution	0.1946
MLE, t-distribution df=2	0.1909
MLE, t-distribution df=10	0.1936

Table 1. R Squared for Different Method

For the third part, I made a table to compare the fitted parameters, which is the beta hat. We found that for the normality assumption of error terms, the intercept is slightly bigger than the t-distribution assumption. On the contrary, the slope from normality assumption is slightly smaller than the t-distribution assumption. For the t-distribution assumptions, when degree of freedom = 2, the intercept is bigger than the degree of freedom = 10. The slope is the opposite. I think these results show that the normal distribution is more sensitive to the outlier (see the above graph of distribution, we have 2 outliers), therefore, the slope is bigger. For t-distribution, since it's a fat tail distribution, it can tolerate more for the outliers.

METHOD	$\hat{\boldsymbol{\beta}}_0$	$\hat{\boldsymbol{\beta}}_1$
OLS, normal distribution	0.1198	0.6052
MLE, normal distribution	0.1198	0.6052
MLE, t-distribution df=2	0.1861	0.5559
MLE, t-distribution df=10	0.1342	0.5639

Table 2. Beta Hat for Different Method

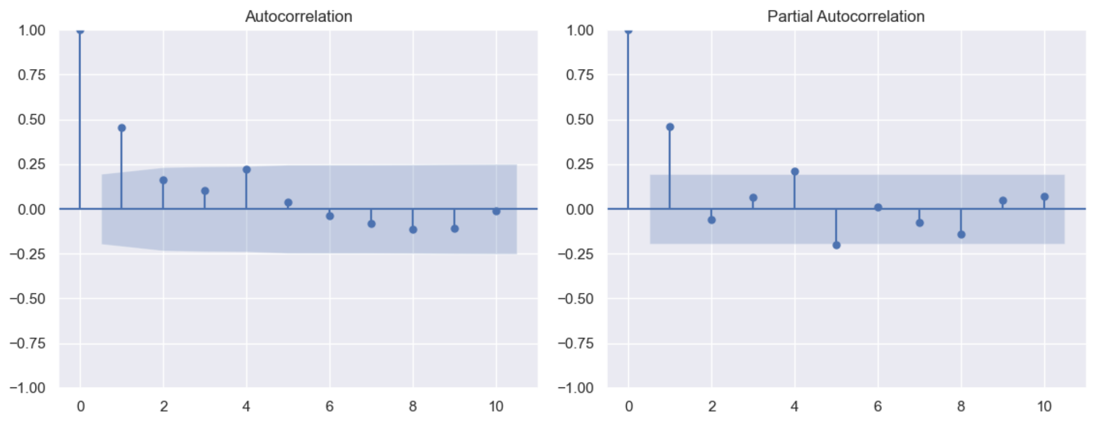
Problem 3

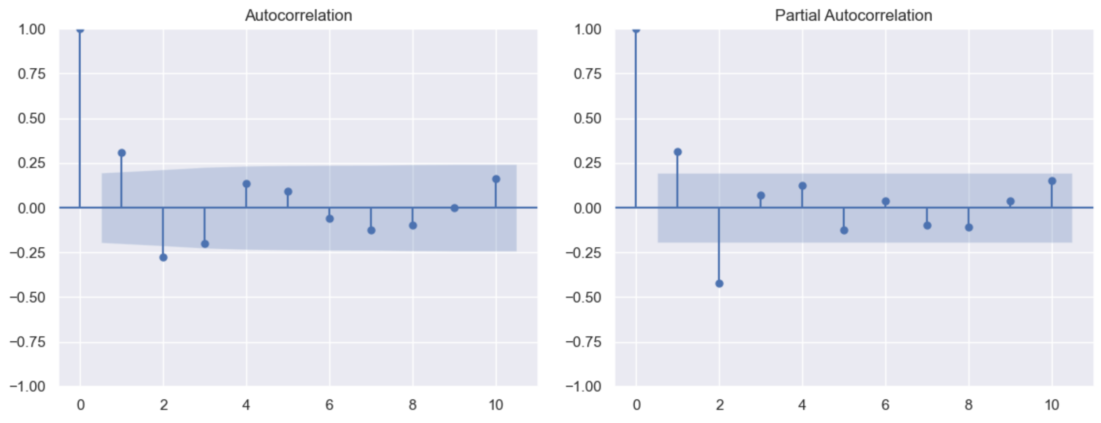
For this problem, basically it asked us to simulate AR() and MA() processes. Then compared the ACF and PACF plots of these processes, and finally asked us to describe how to identify type and order in these graphs.

For simplicity, I decided not to focus (and discussed) intensively on how to choose the parameter in each process since there are multiple way to choose parameters. And the plot here is just one case of them. I will focus more on the explanation part.

For AR(1), I simply choose 0.6 as beta here. For AR(2), the two parameters are 0.6 and -0.3. For AR(3), the three parameters are 0.6, -0.3, 0.2.

And the ACF/PACF plot for AR(), MA() processes are shown below.





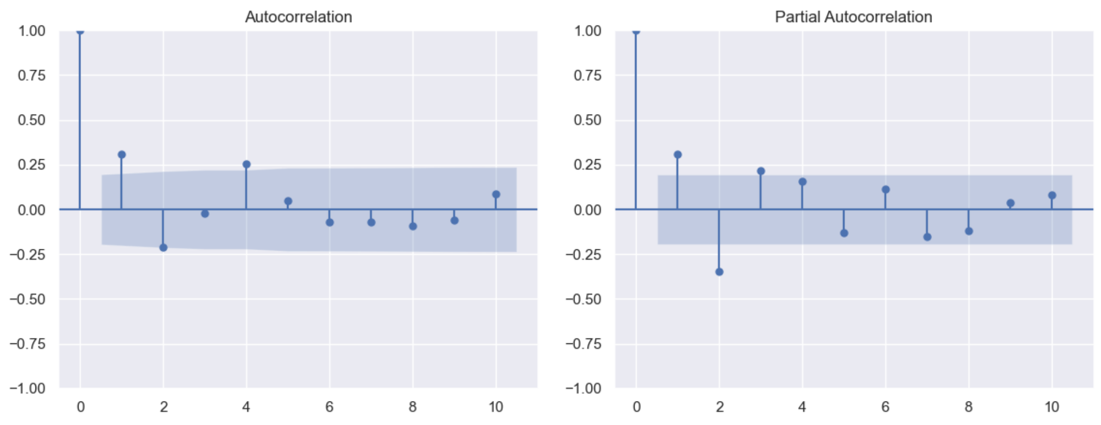


Fig 5. The ACF/PACF Plot of AR(1), AR(2), AR(3)

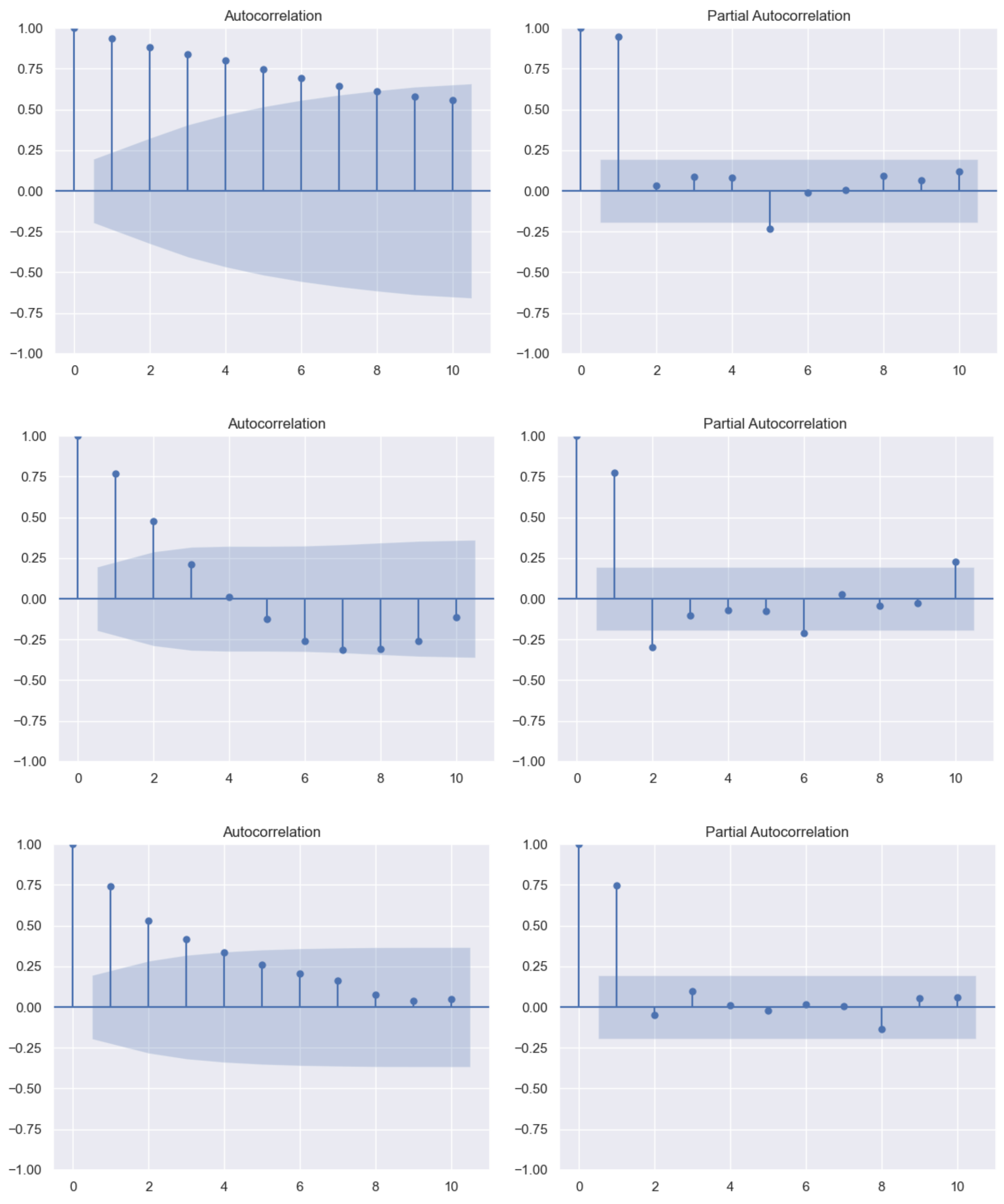


Fig 6. The ACF/PACF Plot of MA(1), MA(2), MA(3)

The ACF plot shows the correlation between a time series and its lagged values. If the ACF plot decays exponentially, it suggests that the time series is an MA model. If the ACF plot has a slow decay or spikes at regular lags, it suggests that the time series is an AR model.

The PACF plot shows the correlation between a time series and its lagged values, while controlling for the effects of any intermediate lags. If the PACF plot decays exponentially, it suggests that the time series is an AR model. If the PACF plot has a slow decay or spikes at regular lags, it suggests that the time series is an MA model.

For determining the order of the AR(), MA() model, in an ACF plot of an AR(p) model, the lag-p autocorrelation will be large and significant, while the autocorrelations at all higher lags will be close to zero. We can also see a "cutoff" pattern.

In a PACF plot of an AR(p) model, the lag-p partial autocorrelation will be large and significant, while the partial autocorrelations at all higher lags will be close to zero. We can also see the "cutoff" pattern here.

By comparing the patterns of the ACF and PACF plots, we can determine the order of the AR model. The first lag at which the PACF plot cuts off is an estimate of the order of the AR model.

We need to also be careful that we need to check the stationarity before we use ACF and PACF. If the process is not stationary, then the ACF and PACF will be useless in helping us determining the order or whether the process is AR() or MA().