Problem 1

I used the random function in the Numpy package to generate a group of random numbers to test if the skew and kurtosis functions are biased.

- First, I generated 100000 random numbers and set mu and sigma as 0 and 0.1 to fit them into the normal distribution. The skewness of this group of number is -0.00373.
- Second, I created a for loop to calculate the skewness of 100000 random numbers 10000 times and append the result into an empty list.
- Third, I calculated the mean and standard deviation of the list and got results of -6.46553 and 0.00774. I also calculated the t-statistics and the p-value. I had ran this step for a number of times, and for most of them, the p-value is greater than the 0.5 critical value.

I did the same steps once again for kurtosis function. I finally got a mean of -6.00697 and standard deviation 0.01545. I also ran the program few time to test the p-value. The result is that for most of the time p-value is greater than 0.5.

However, I don't think that the results stated above are valid enough. I tried the above procedures one more time for skewness and kurtosis with a smaller sample size of 100. The result for skewness didn't change too much: I got a large p-value to conclude that it is hard to prove that it is biased. On the other hand, the p-value for kurtosis became very small, like 5.41205*(10^-53). So, I tend to conclude that kurtosis is biased in small samples.

Problem 2

I firstly fit the data using OLS function in statsmodels.api package.

OLS Regression Results

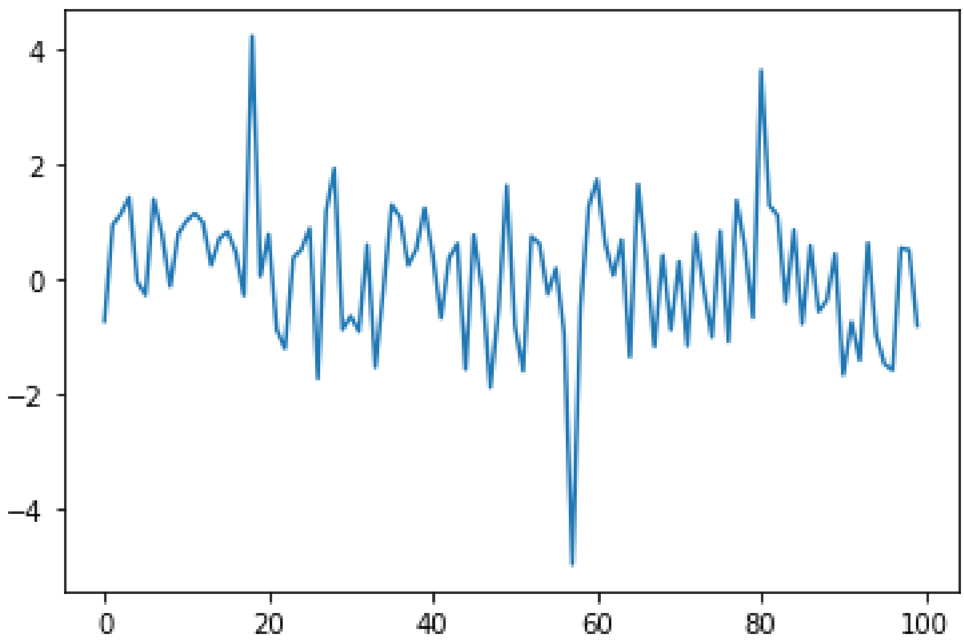
Dep. Variable:	y	**R-squared (uncentered):**	0.193
Model:	OLS	**Adj. R-squared (uncentered):**	0.185
Method:	Least Squares	**F-statistic:**	23.69
Date:	Sat, 28 Jan 2023	**Prob (F-statistic):**	4.28e-06
Time:	05:06:35	**Log-Likelihood:**	-160.49
No. Observations:	100	**AIC:**	323.0
Df Residuals:	99	**BIC:**	325.6
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
x	0.6052	0.124	4.867	0.000	0.358	0.852

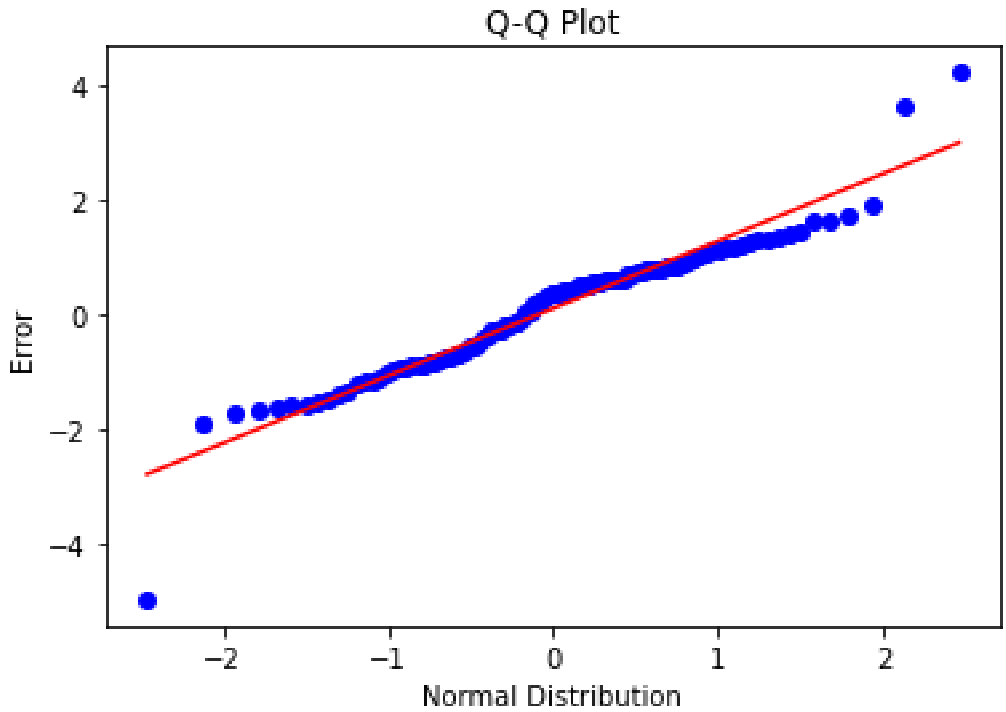
Omnibus:	14.146	**Durbin-Watson:**	1.866
Prob(Omnibus):	0.001	**Jarque-Bera (JB):**	43.674
Skew:	-0.267	**Prob(JB):**	3.28e-10
Kurtosis:	6.193	**Cond. No.**	1.00

Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Since I have the coefficient of x which is 0.6052, I could get the error vector using $\epsilon = Y - X\beta$. Here is a plot of the errors.



To determine whether the errors fit the normal distribution, I choose to use a Q-Q plot. As shown on the plot, the blue dots, which represents errors, mostly follow the pattern of the standard red line. So, errors fit well to normal distribution.



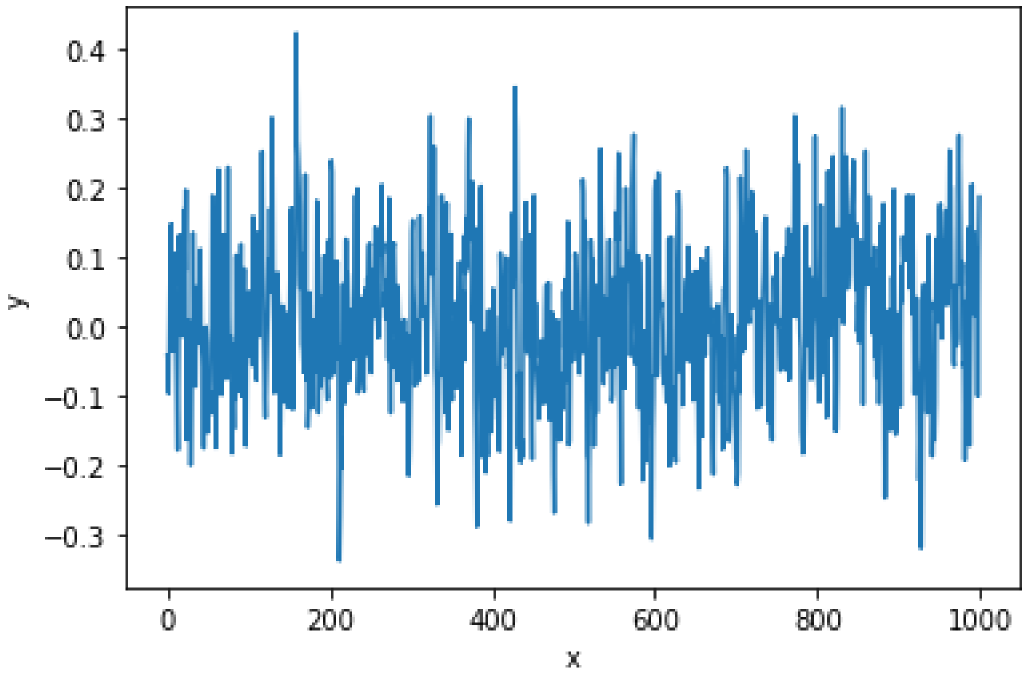
Then I would fit the data to MLE based on the assumption of normal or t distribution of the errors. To find the optimized beta, I calculated the negative log-likelihood for each situation and minimize it. The beta of normal distribution is 0.6052 (just as OLS indicated), and that of t distribution is 0.96491.

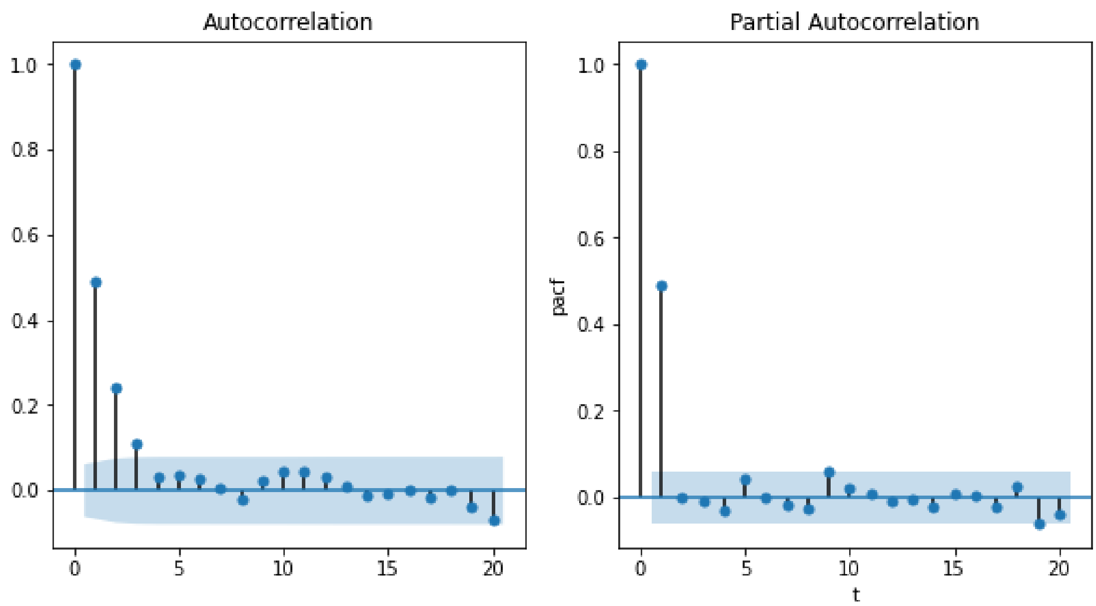
One method to compare these two assumptions is AIC, which equals to $2k - 2log(L\hat{})$. AIC of normal distribution is 325.98419 and that of t distribution is 284.44006. Hence, the errors have a better fit to t distribution. However, assumption of normality generates the same beta as the OLS model, so the breaking of normality might not be able to give us the best fit model.

Problem 3

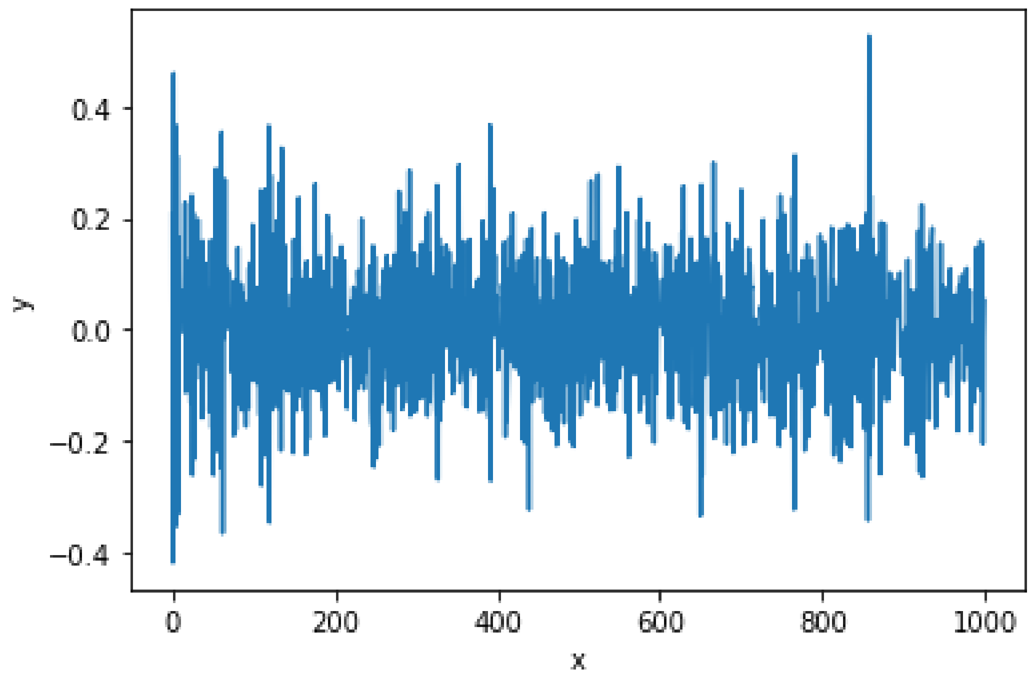
I used arma_generate_sample in statsmodels.tsa.arima_process to simulate the AR and MA.

AR(1):

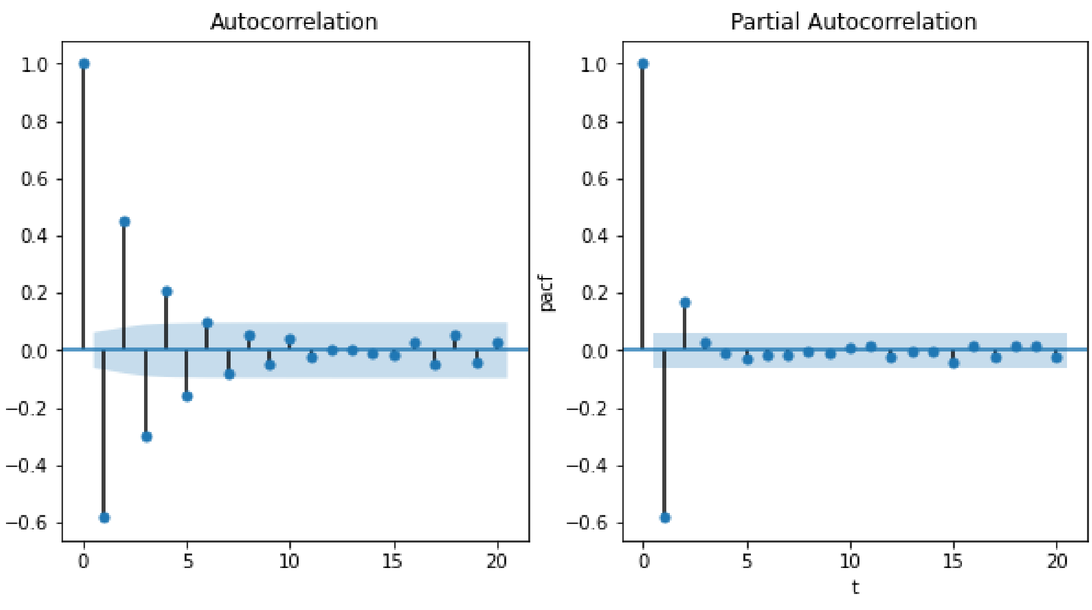




AR(2):

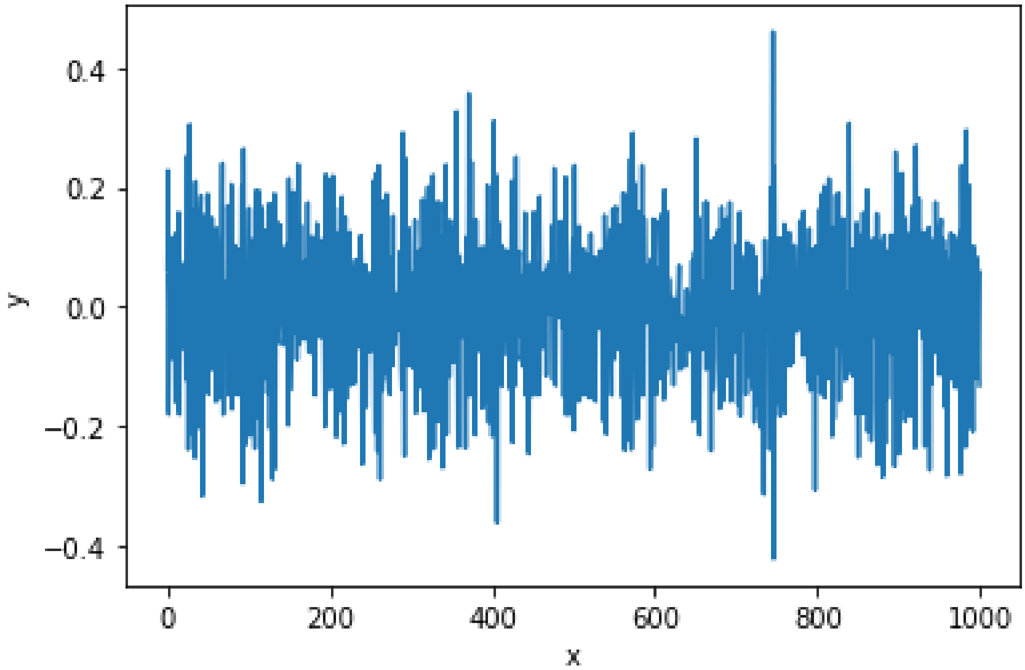


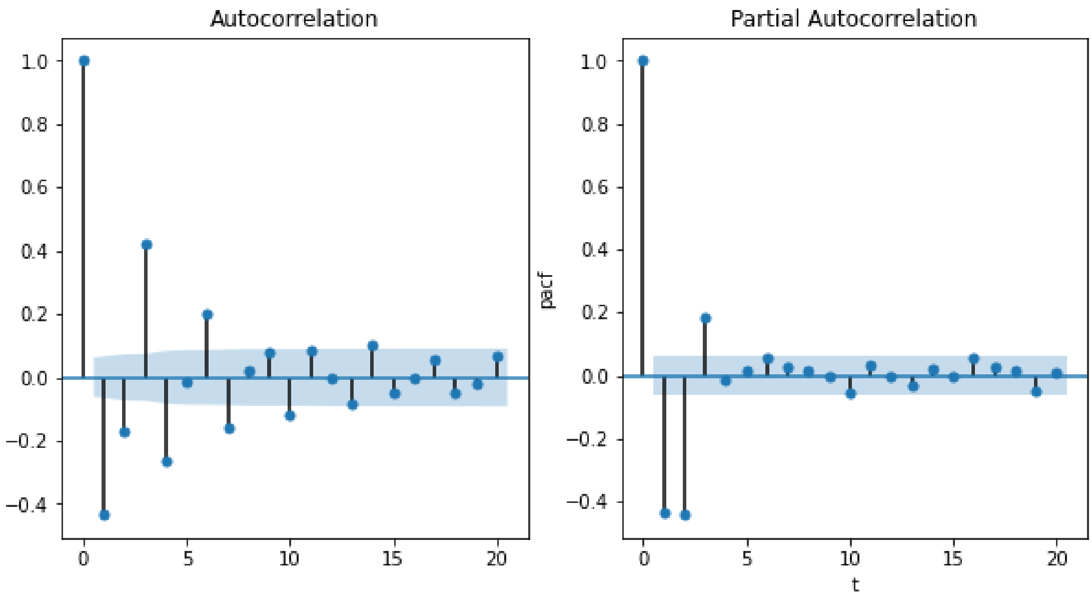
0.4
0.2
0.0
−0.2
−0.4
y
0
200
400
600
800
1000
x



Autocorrelation
Partial Autocorrelation
pacf
t

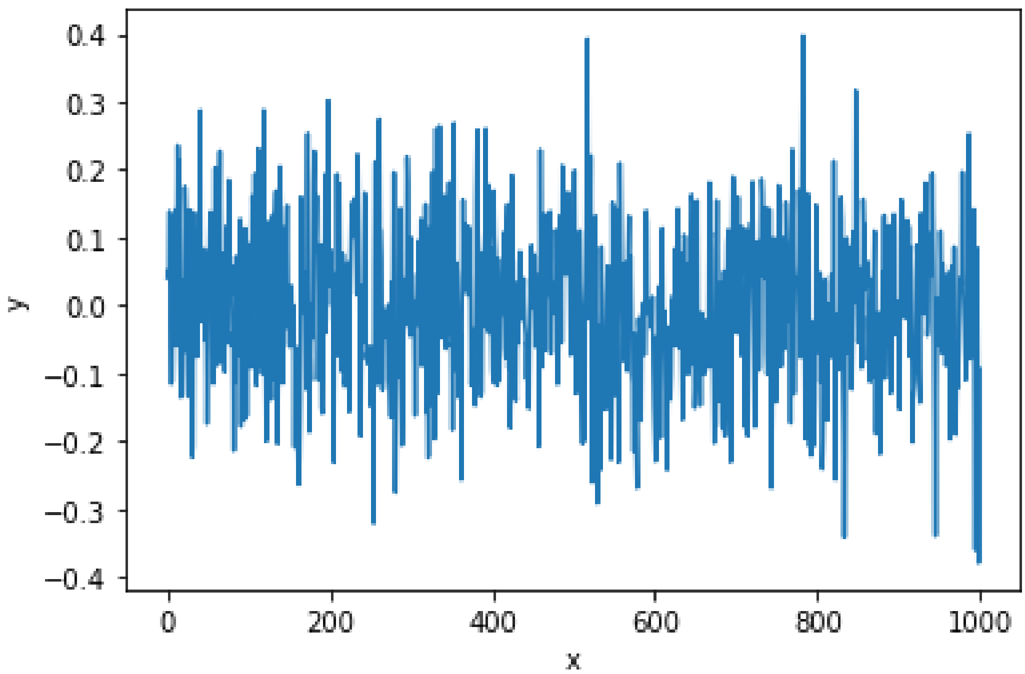
AR(3):



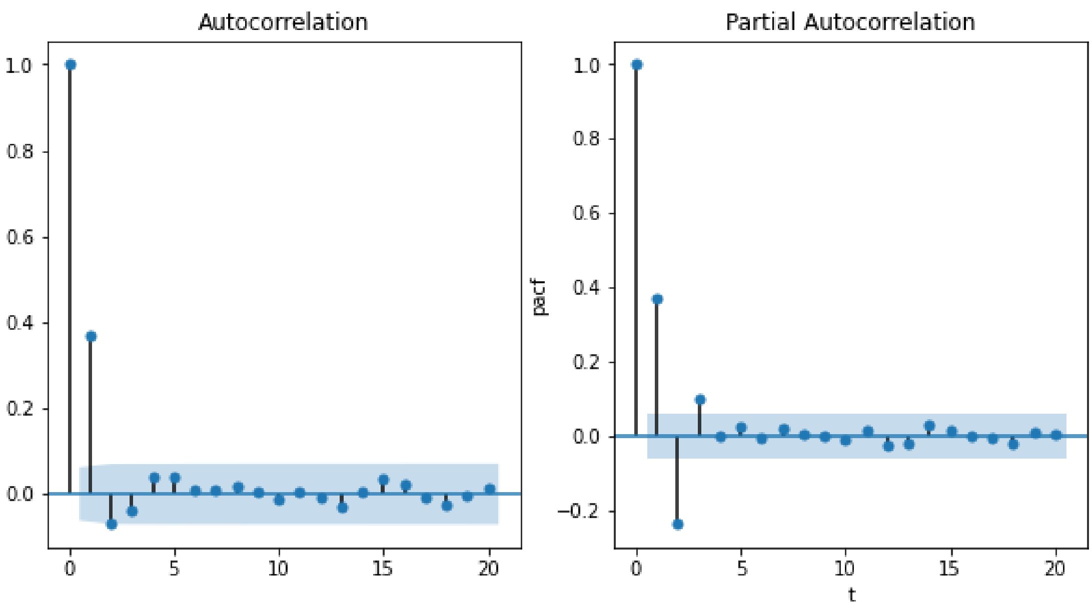


For AR(p) processes, the PACF will have non-zero significant coefficients (points outside the light blue area) for the first p lags and will taper off for the lags beyond that. The ACF will also have large non-zero significant coefficients for the first p lags, but will also have significant values for higher lags.

MA(1):

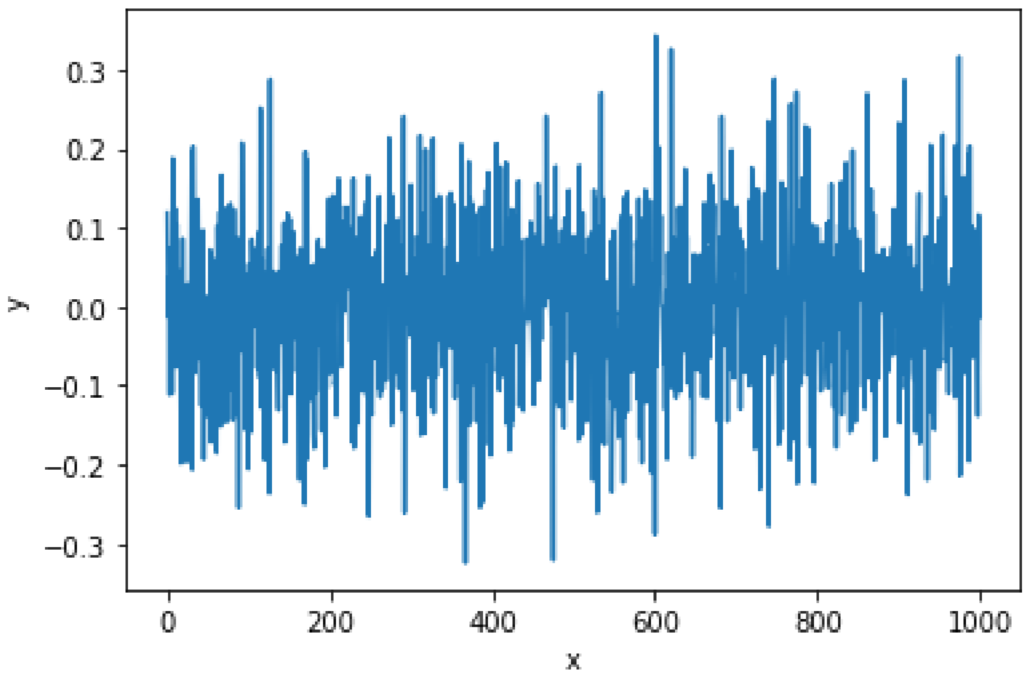


0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
y
0
200
400
600
800
1000
x

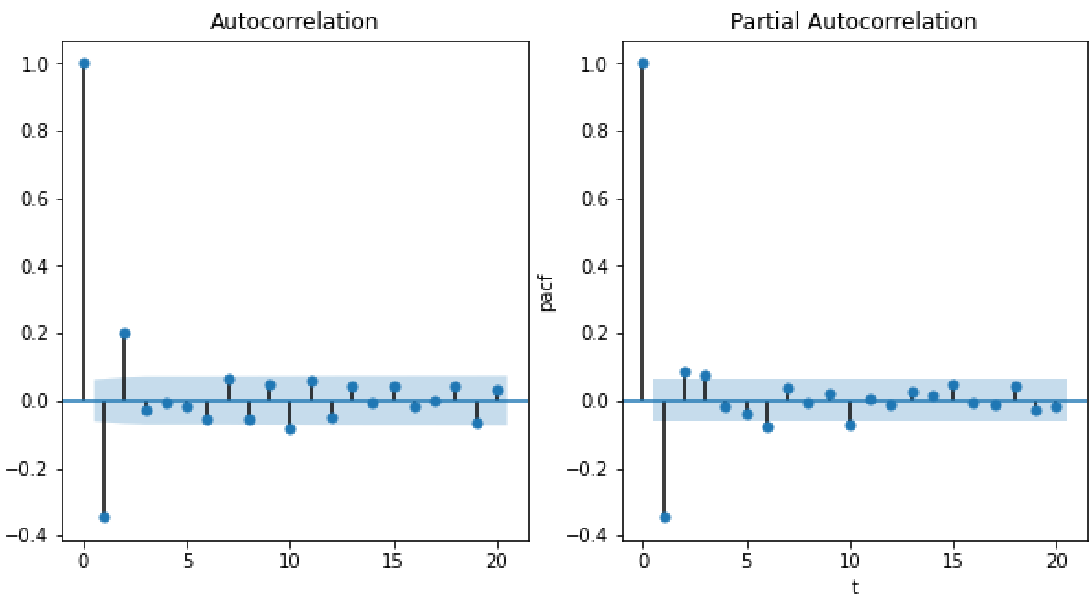


Autocorrelation
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
Partial Autocorrelation
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
pacf
0
5
10
15
20
t

MA(2):

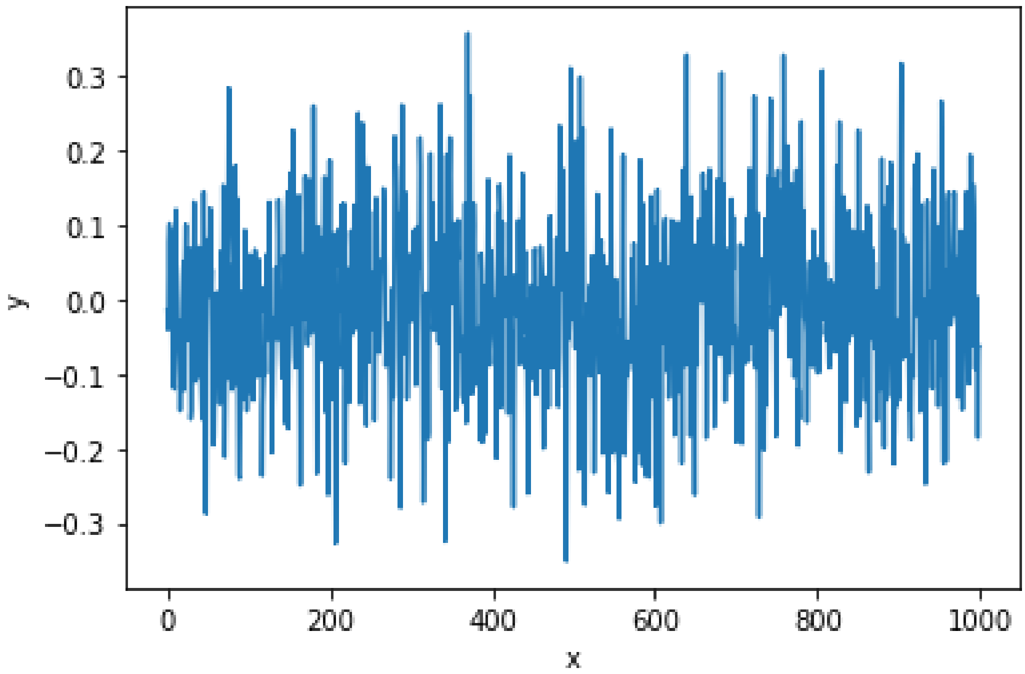


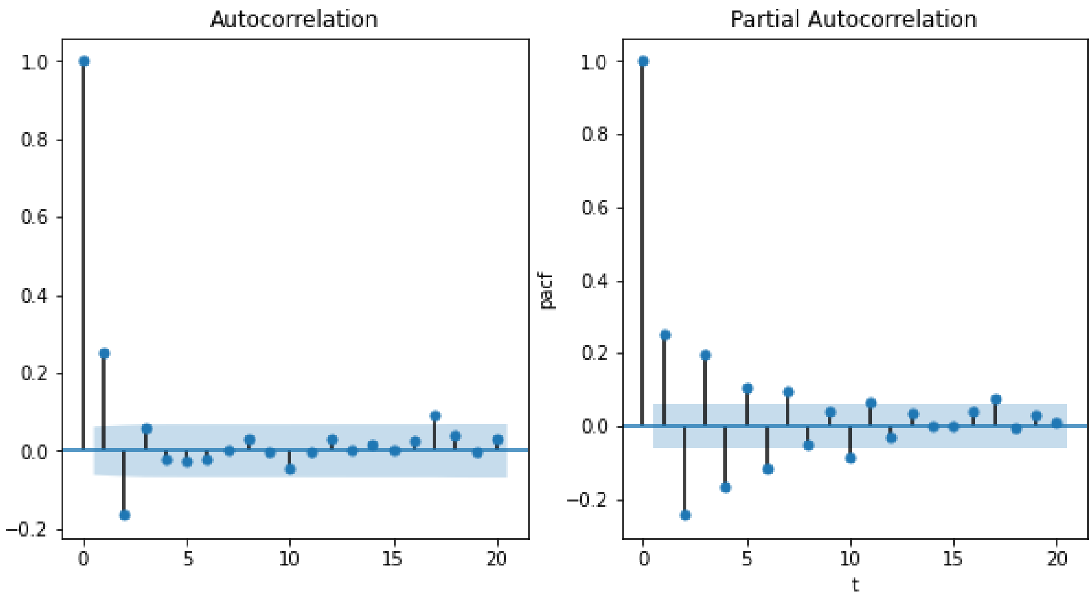
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
y
0
200
400
600
800
1000
x



Autocorrelation
Partial Autocorrelation
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
pacf
0
5
10
15
20
t

MA(3):





For MA(q) processes, the PACF will have non-zero significant coefficients for the first q lags and will quickly taper off after that. The ACF is a little bit vague to identify. It seems to have spikes at the q-th and higher lags.