Xinyi (Michelle) Wan

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.

In Python, scipy.stats.skew and scipy.stats.kurtosis calculate the skewness and excess kurtosis of a dataset. We hypothesize that they return unbiased estimators of the population. To test our hypothesis, we can run Monte Carlo simulations using the functions and then see if the Monte Carlo estimates are statistically different from the population parameters. The detailed steps are as follows.

1. Choose a sample size N. Generate K random samples of size N from a normal distribution (whose skewness and excess kurtosis are both zero).
2. Calculate the skewness (excess kurtosis) of each sample using stats.skew (stats.kurtosis).
3. Use one sample t-test to determine whether the mean of the skewness (excess kurtosis) values is statistically different from zero. Again, the null hypothesis is that stats.skew (stats.kurtosis) returns an unbiased estimator, and the alternative hypothesis is that it returns a biased estimator.
4. Repeat 1-3 with different values of N and K.

A thousand samples of size 10, 1000, and 5000 were drawn, and we found that we have sufficient evidence to reject the hypothesis of unbiased stats.kurtosis when the sample sizes are relatively small. However, as the sample size increases, we fail to reject the hypothesis of unbiased stats.kurtosis. Regarding skewness, we fail to reject the hypothesis of unbiased stats.skew for both small and large sample sizes. I then increased the sample number to K=10000 with the hope to get a more accurate conclusion, which turned out to be the same as when K=1000. The result table is as below.

Sample number, K	**Sample Size, N**	**H_0: unbiased stats.skew**		**H_0: unbiased stats.kurtosis**	
		p-value (at $\alpha = 0.05$)	Reject H_0?	p-value (at $\alpha = 0.05$)	Reject H_0?
1000	10	0.801	Fail to reject	0.000	Reject
	1000	0.206	Fail to reject	0.004	Reject
	5000	0.494	Fail to reject	0.057	Fail to reject
	10000	0.719	Fail to reject	0.471	Fail to reject
10000	10	0.423	Fail to reject	0.000	Reject
	1000	0.705	Fail to reject	0.000	Reject
	5000	0.144	Fail to reject	0.131	Fail to reject
	10000	0.196	Fail to reject	0.628	Fail to reject

In order to further investigate whether stats.skew is biased, we can draw a lot of samples (K=100 0000) from a normal distribution and for each sample, we compare the **bias-uncorrected** skewne ss calculated using formula (1) to the skewness we get from stats.skew. Through this process, the maximum difference obtained is 1.9984014443252818e-15 and the minimum difference is -1.99 84014443252818e-15. The differences are so close to zero that stats.skew is likely biased, but it cannot be proven statistically.

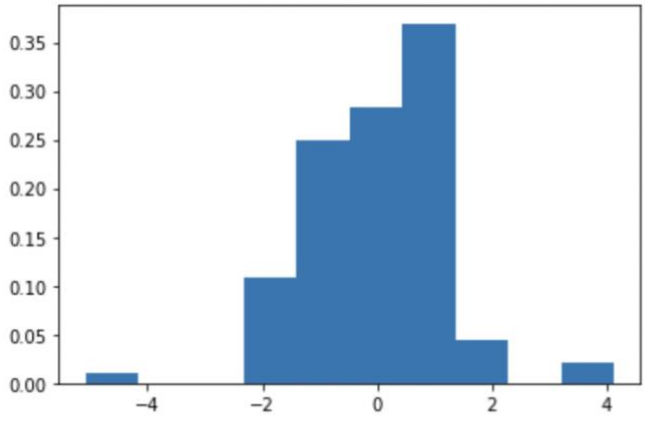
$$\frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\overline{x})^4}{\left[\frac{1}{n}\sum_{i=1}^{n}(x_i-\overline{x})^2\right]^2}-3 \qquad (1)$$

In a nutshell, although stats.kurtosis may become less biased as the sample size increases, it is biased for small sample sizes. On the other hand, stats.skew is likely to be biased but it cannot be proven statistically. These conclusions are consistent with the fact that stats.skew and stats.kurtosis both have a “bias” parameter, which is set to True by default.

Problem 2

Q1. 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?

In the context of OLS, the error vector and residuals are the same and used interchangeably. By fitting the data using statsmodels’s OLS function, we can obtain the intercept and the coefficient of the fit and the residuals. I then plotted the distribution of the residuals and calculated its skewness and excess kurtosis. The results are as follows.




```
Mean of error vector: -1.6653345369377347e-17
Variance of error vector: 1.436148485406261
Skewness of error vector: -0.271353974448504
Excess kurtosis of error vector: 3.4211883977033253
ols fit params:
const    0.119836
x        0.605205
```

Even with the naked eye, one can tell that the distribution deviates considerably from a normal distribution. A skewness of -0.271 and an excess kurtosis of 3.421 further validate this.

Q2. 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?

The target is to find parameters of a fit that maximizes the joint probability of the observed errors, given an assumption of errors' distribution. We can approach the problem by minimizing the negative log-likelihood function (which is equivalent to maximizing the log-likelihood function) that conforms to the normality assumption and the assumption of a t-distribution of errors, respectively.

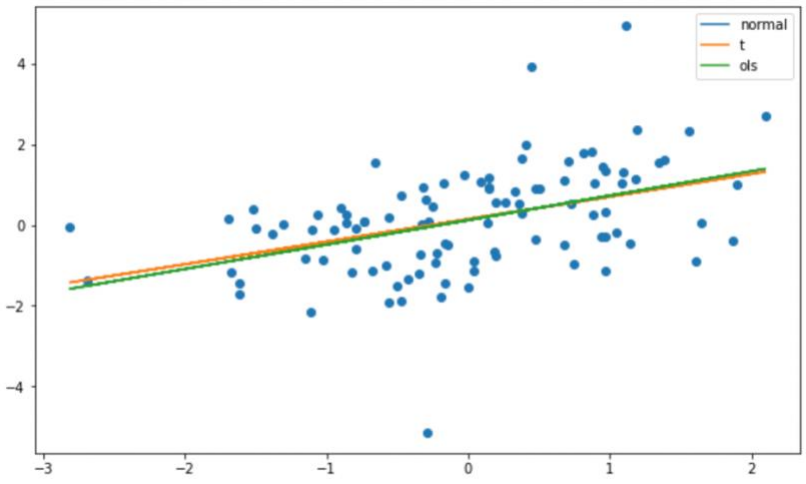
The log-likelihood of the t-distribution can be obtained through stats.t.logpdf, and the log-likelihood of the normal distribution is given by

$$ll = -\frac{n}{2}ln\left(\sigma^2 2\pi\right) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(x_i - \mu\right)^2$$

It is worth noting that the maximization of the above function is equivalent to maximizing the log-likelihood of the error terms with $\mu = 0$, which is

$$ll = -\frac{n}{2}ln\left(\sigma^2 2\pi\right) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(\epsilon_i - 0\right)^2$$

This then explains why fitting the data using MLE given the assumption of normality and fitting the data in the context of OLS render the same parameter results. OLS tries to minimize the sum of squared residuals, and at the same time, MLE given the normality assumption also only considers the second part of the above log-likelihood function, as the first part, $\frac{n}{2}ln\left(\sigma^2 2\pi\right)$, is constant with respect to μ and does not affect the estimates in the linear regression context.



This plot shows that the OLS fit and the MLE fit given the normality assumption overlap. To evaluate the goodness of fits, adjusted R-squared, AIC, and BIC were calculated. The table below shows the parameters of the two fits and the respective evaluation metrics.

	Normal distribution	**t-distribution**
Intercept	0.1198	0.1426
Coefficient of x	0.6052	0.5576
R-squared	0.1946	0.1931
AIC	325.98	316.95
BIC	333.80	324.76

The R-squared of the fit given the assumption of a t-distribution of errors is only marginally smaller than that of the fit given the normality assumption, but both AIC and BIC values of the former are much smaller than their counterparts. Therefore, we conclude that the fit under the assumption of a t-distribution of the errors is a better fit.

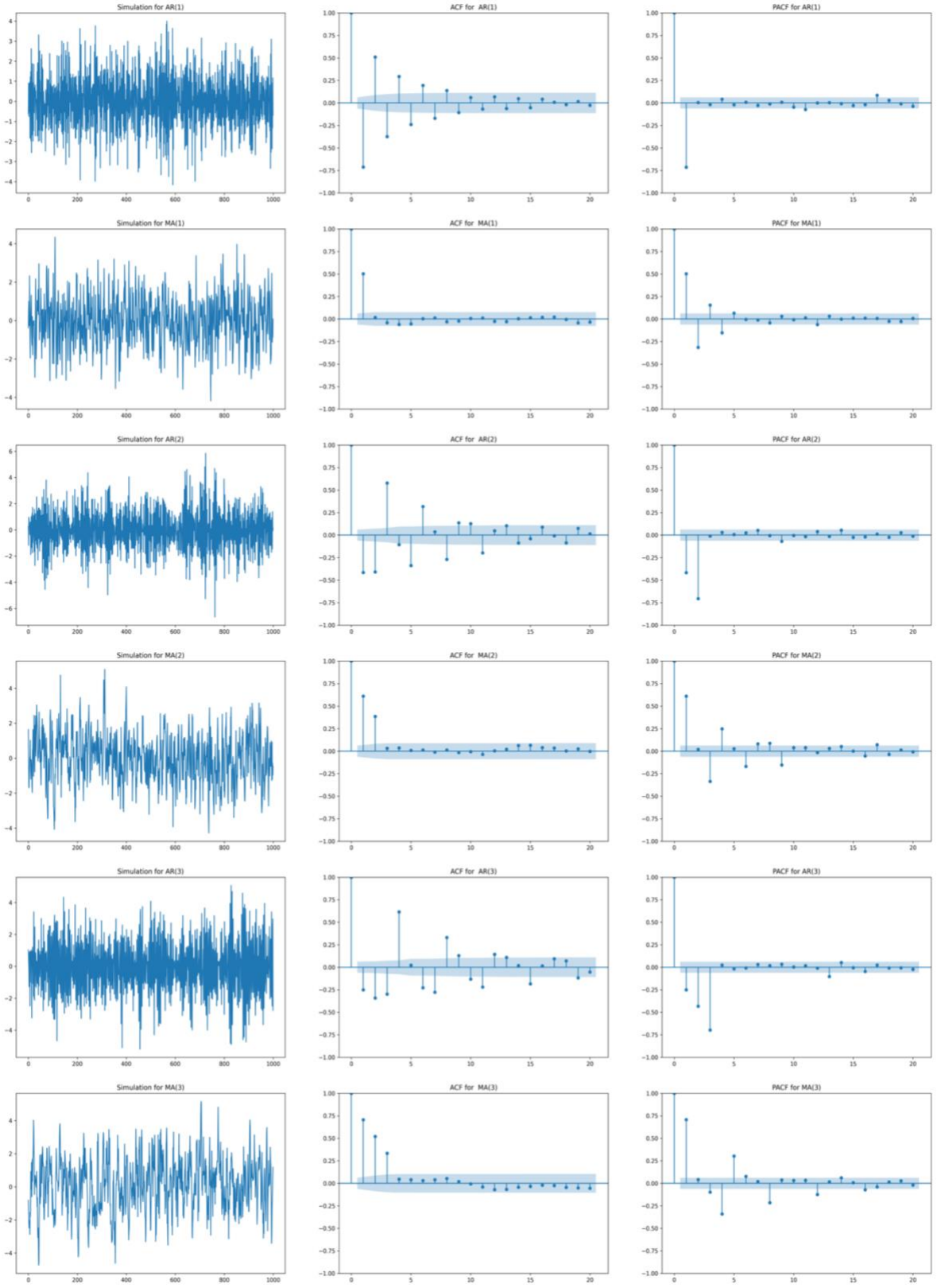
Q3. What are the fitted parameters of each and how do they compare? What does this tell us about the breaking of the normality assumption in regard to expected values in this case?

The fitted parameters of each and the plot of the fits are shown above. They are slightly different. In this case, where the normality assumption is broken, OLS does not fit as well as the MLE fit does under the assumption of a t-distribution of errors. The expected values given by the OLS fit could be satisfactory, but are less accurate nonetheless as suggested by the lower AIC and BIC.

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?

We can simulate AR and MA processes using tsa.arima_process.arma_generate_sample from statsmodels, which takes in the coefficients for AR and MA lag polynomials. The results (with coefficient 0.7) are on the next page. We can tell from the graphs that an ACF showing oscillating values that progressively decay and a PACF that is significant for only the first few lags indicate an AR process, the order of which equals the number of significant lags of PACF (excluding zero lag). On the other hand, a PACF showing oscillating values that gradually decay and an ACF that is significant for only the first few lags indicate an MA process, the order of which equals the number of significant lags of ACF (excluding zero lag).


Simulation for AR(1)
ACF for AR(1)
PACF for AR(1)
Simulation for MA(1)
ACF for MA(1)
PACF for MA(1)
Simulation for AR(2)
ACF for AR(2)
PACF for AR(2)
Simulation for MA(2)
ACF for MA(2)
PACF for MA(2)
Simulation for AR(3)
ACF for AR(3)
PACF for AR(3)
Simulation for MA(3)
ACF for MA(3)
PACF for MA(3)