Response

Problem 1

The conditional distribution of the Multivariate Normal to the OLS equations are the same.

Mathematical Proof

From OLS, we have

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i \tag{1}$$

where

$$\hat{\beta}_1 = \frac{\sigma_{xy}}{\sigma_{x^2}} \tag{2}$$

Taking expectation on both sides, we have

$$E(Y) = E(\hat{\beta}_0 + \hat{\beta}_1 X) = E(\hat{\beta}_0) + E(\hat{\beta}_1 X) = \hat{\beta}_0 + \hat{\beta}_1 E(X)$$

Which is

$$\bar{Y} = \hat{\beta}_0 + \hat{\beta}_1 \bar{X}$$

Therefore, we have

$$\hat{\beta}_0 = \bar{Y} - \hat{\beta}_1 \bar{X} \tag{3}$$

From the conditional expectation of conditional bivariate normal, we have

$$E(Y|X = x_i) = E(Y) + \frac{\sigma_{xy}}{\sigma_{x^2}}(x_i - E(X))) \tag{4}$$

Substitute (2), (3) in (1), we have

$$\hat{y_i} = (\bar{Y} - \hat{\beta}_1\bar{X}) + \hat{\beta}_1 x_i$$

$$\hat{y_i} = \bar{Y} + (\hat{\beta}_1 x_i - \hat{\beta}_1\bar{X})$$

$$\hat{y_i} = \bar{Y} + \hat{\beta}_1(x_i - \bar{X})$$

$$\hat{y_i} = \bar{Y} + \frac{\sigma_{xy}}{\sigma_{x^2}}(x_i - \bar{X}) \tag{5}$$

Which is the same as (4).

Empirical proof

Empirically, we construct the conditional expectation of Y in the following code

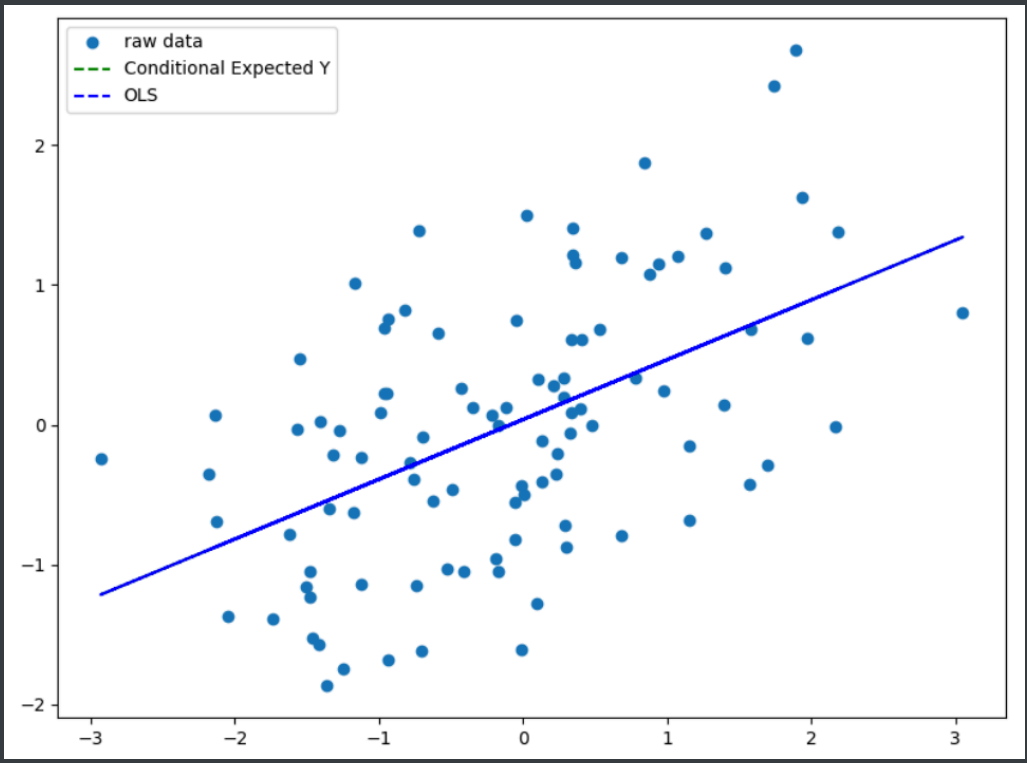
```
# construct vector x and vector y
vectorX = data.iloc[:, 0]
vectorY = data.iloc[:, 1]

# calculate expectation of x and y
meanX = sum(vectorX)/len(vectorX)
meanY = sum(vectorY)/len(vectorY)

# calculate covariance matrix
covZ = np.cov(vectorX, vectorY, ddof=1)
covXX = covZ[0, 0]
covXY = covZ[0, 1]
covYX = covXY
covYY = covZ[1, 1]

# Expectation of Y given X
conExpY = pd.Series([meanY + covYX / covXX * (x - meanX) for x in
vectorX])
```

and estimate the OLS regression.

```
result = sm.ols(formula="y~x", data=data).fit()
fittedY = result.fittedvalues
print(result.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.268
Model:                            OLS   Adj. R-squared:                  0.261
Method:                 Least Squares   F-statistic:                     35.89
Date:                Fri, 14 Jan 2022   Prob (F-statistic):           3.47e-08
Time:                        23:40:12   Log-Likelihood:                -120.46
No. Observations:                 100   AIC:                             244.9
Df Residuals:                      98   BIC:                             250.1
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.0379      0.082      0.461      0.646      -0.125       0.201
x              0.4280      0.071      5.990      0.000       0.286       0.570
==============================================================================
Omnibus:                        5.101   Durbin-Watson:                   2.006
Prob(Omnibus):                  0.078   Jarque-Bera (JB):                2.716
Skew:                           0.145   Prob(JB):                        0.257
Kurtosis:                       2.246   Cond. No.                         1.20
==============================================================================
```

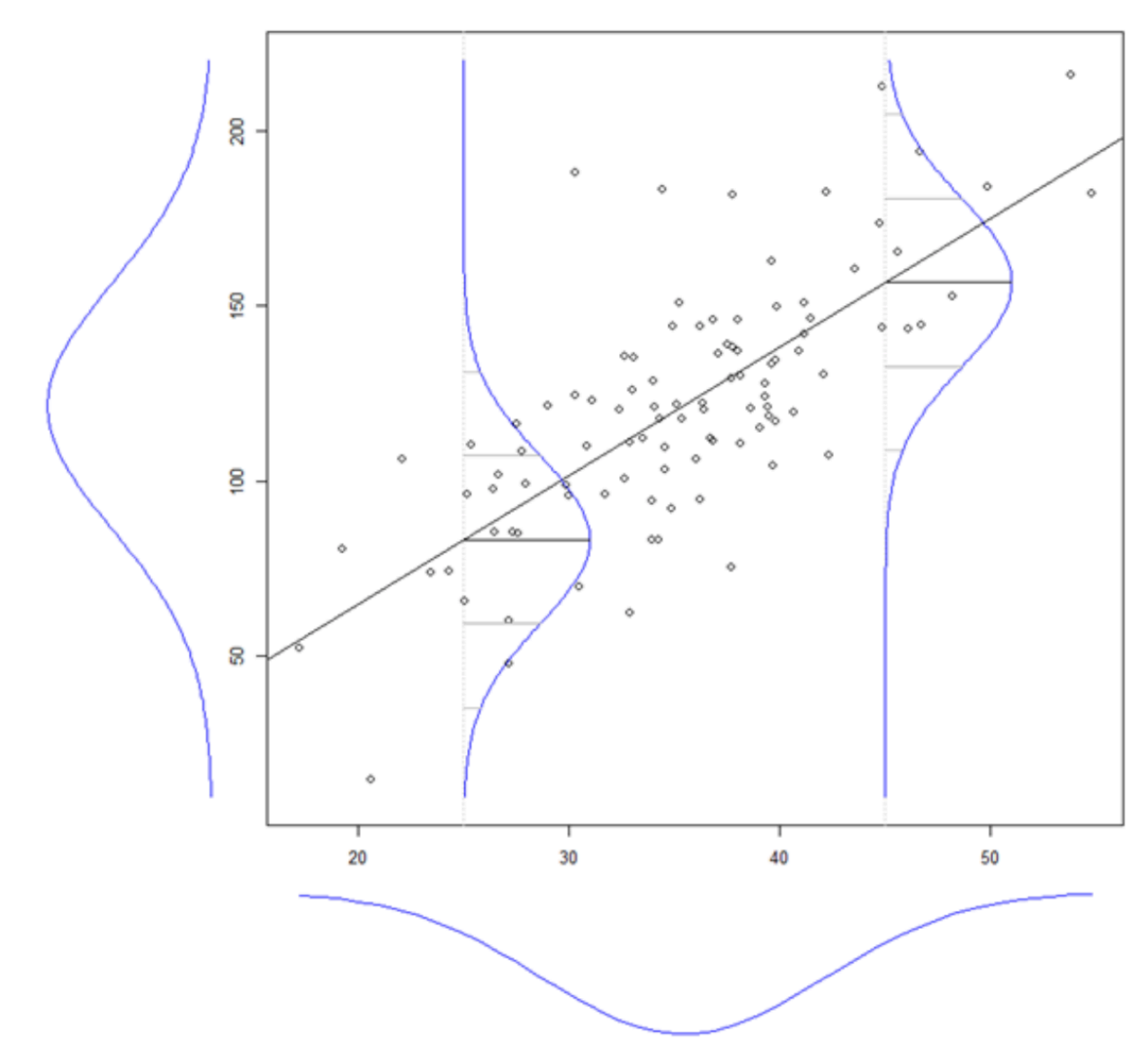
And plot the conditional expected Y and fitted Y in the same graph



They are exactly the same.

Why?

OLS is to estimate Y given X, which has the same meaning of finding the conditional distribution of Y given X. If the covariance between X and Y is not 0, the mean of the conditional distribution of Y has to adjust according to the covariance between X and Y.

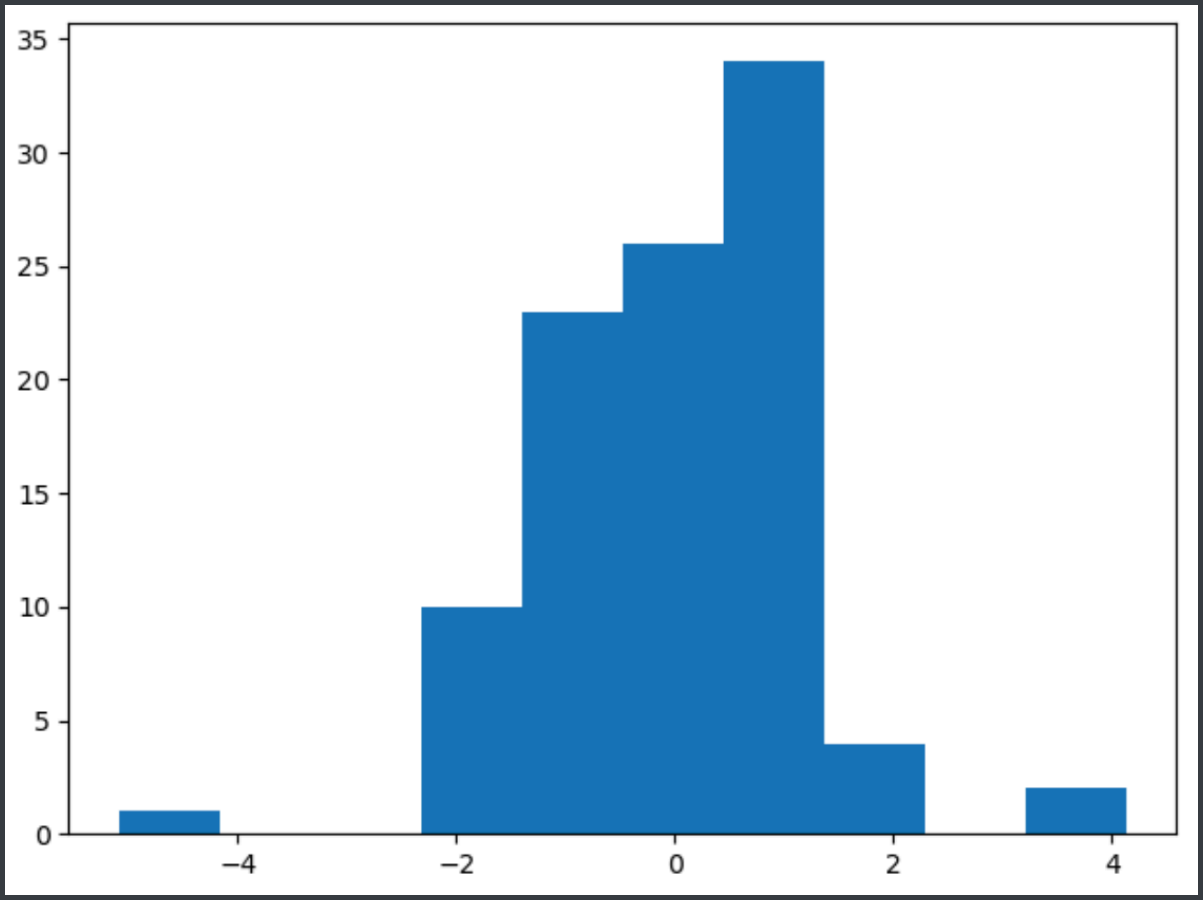


This graph vividly shows the reason why the two are the same. The graph shows that the margin distribution of X and Y are both normal. Also, X and Y are positively correlated. the fitted Y given X=x0 is exactly the mean of Y's normal distribution given X=x0. The conditional mean adjusts when X changes.

Problem 2

OLS

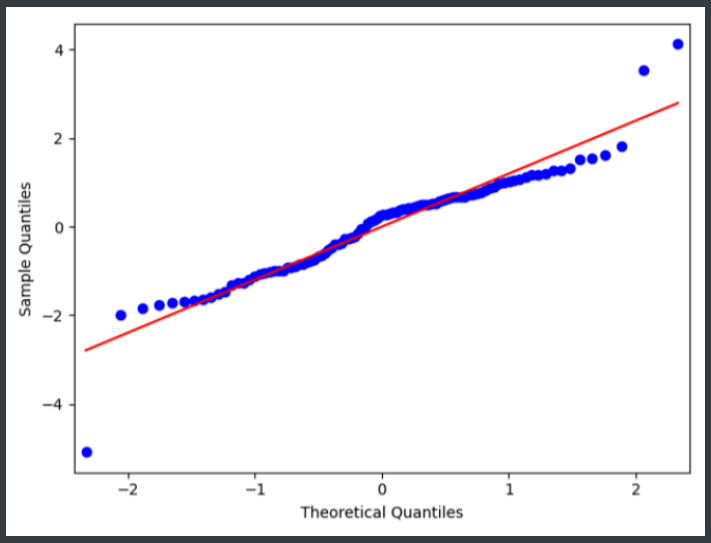
Distribution of the error vector using OLS



How well does it fit the assumption of normally distributed errors?

To know how well it fits the assumption of normally distributed errors, I use qqplot and SW test to estimate its normality. qqplot shows the distribution of the data against the expected normal distribution. For normally distributed data, observations should lie approximately on a straight line. Shapiro-Wilk test is a test of normality.

- qqplot



- SW test

```
Statistics=0.938, p=0.000
Sample does not look Gaussian (reject H0)
```

As seen in the pictures, the histogram of the error vector is skewed, the qqplot is not straight, and the SW test reject the hypothesis that the Sample is an Gaussian.

Therefore, it does not fit the assumption of normally distributed errors.

MLE

MLE with normal assumption and t assumption

To get the MLE with normal assumption, I first define a negative log likelihood function by passing in the normal distribution with mean 0 and initial betas and standard deviation of the error term, then take log and sum each term of it. Then, I minimize this function with method BFGS.

```
def negLogLikeForNormal(parameters):
    const, beta, std_dev = parameters
    e = data['y'] - const - beta * data['x']
    return -1 * stats.norm(0, std_dev).logpdf(e).sum()


paramsNormal = np.array([1, 1, 1])
resNormal = minimize(negLogLikeForNormal, paramsNormal, method='BFGS')
data['mleResidualNormal'] = data['y'] - (resNormal.x[0] + resNormal.x[1]
* data['x'])
```

To get the MLE with generalized t assumption, I first define a negative log likelihood function by passing in the t distribution with initial betas, df, and scale, then take log and sum each term of it. Then, I minimize this function with method BFGS.

```
def negLogLikeForT(parameters):
    const, beta, df, scale = parameters
    e = data['y'] - const - beta * data['x']
    return -stats.t(df=df, scale=scale).logpdf(e).sum()


paramsT = np.array([1, 1, 1, 1])
resT = minimize(negLogLikeForT, paramsT, method='BFGS')
data['mleResidualT'] = data['y'] - (resT.x[0] + resT.x[1] * data['x'])
```

Goodness of fit and Information criterion

To see which assumption fits the best, I try the R square, AIC, and BIC.

R square is calculated by

$$R^2 = 1 - \frac{SS_{error}}{SS_{total}}$$

AIC is calculated by

$$AIC = 2k + 2(-ln\hat{L})$$

And BIC

$$BIC = 2(-ln\hat{L}) + d * ln(N)$$

In code

```
# goodness of fit
ssTotal = ((data['y'] - np.mean(data['y']))**2).sum()
ssErrorNormal = (data['mleResidualNormal']**2).sum()
ssErrorT = (data['mleResidualT']**2).sum()
rSquareNormal = 1 - ssErrorNormal/ssTotal
rSquareT = 1 - ssErrorT/ssTotal

# Information Criteria
aicNormal = 2 * len(resNormal.x) + 2 * negLogLikeForNormal(resNormal.x)
aicT = 2 * len(resT.x) + 2 * negLogLikeForT(resT.x)

bicNormal = 2 * negLogLikeForNormal(resNormal.x) + len(resNormal.x) *
np.log(len(data))
bicT = 2 * negLogLikeForT(resT.x) + len(resT.x) * np.log(len(data))
```

The result is

```
R square for MLE with normal distributed errors:  0.19463952391894945
R square for MLE with t distributed errors:  0.19314272628953366
AIC for normal distributed error:  325.98419337832524
BIC for normal distributed error:  333.7997039362895
AIC for T distributed error:  318.94594082493705
BIC for T distributed error:  329.3666215688894
```

Lower AIC/BIC and higher R square indicate better models. Therefore, The MLE using the assumption of a normal distribution of the errors is the best fit.

What are the fitted parameters of each and how do they compare?

The fitted parameters of OLS are the same as the fitted parameters of MLE with normal assumption.

The fitted parameters of MLE with normal assumption are different from those of the MLE with t assumption.

```
OLS parameters:
Intercept 0.119836
Beta     0.605205

MLE with normal assumption:
Intercept  0.11983618971823155
beta  0.6052048664301799


MLE with t assumption:
Intercept  0.14261413186614416
beta  0.5575716616325362
```

What does this tell us about the breaking of the normality assumption in regards to expected values in this case?

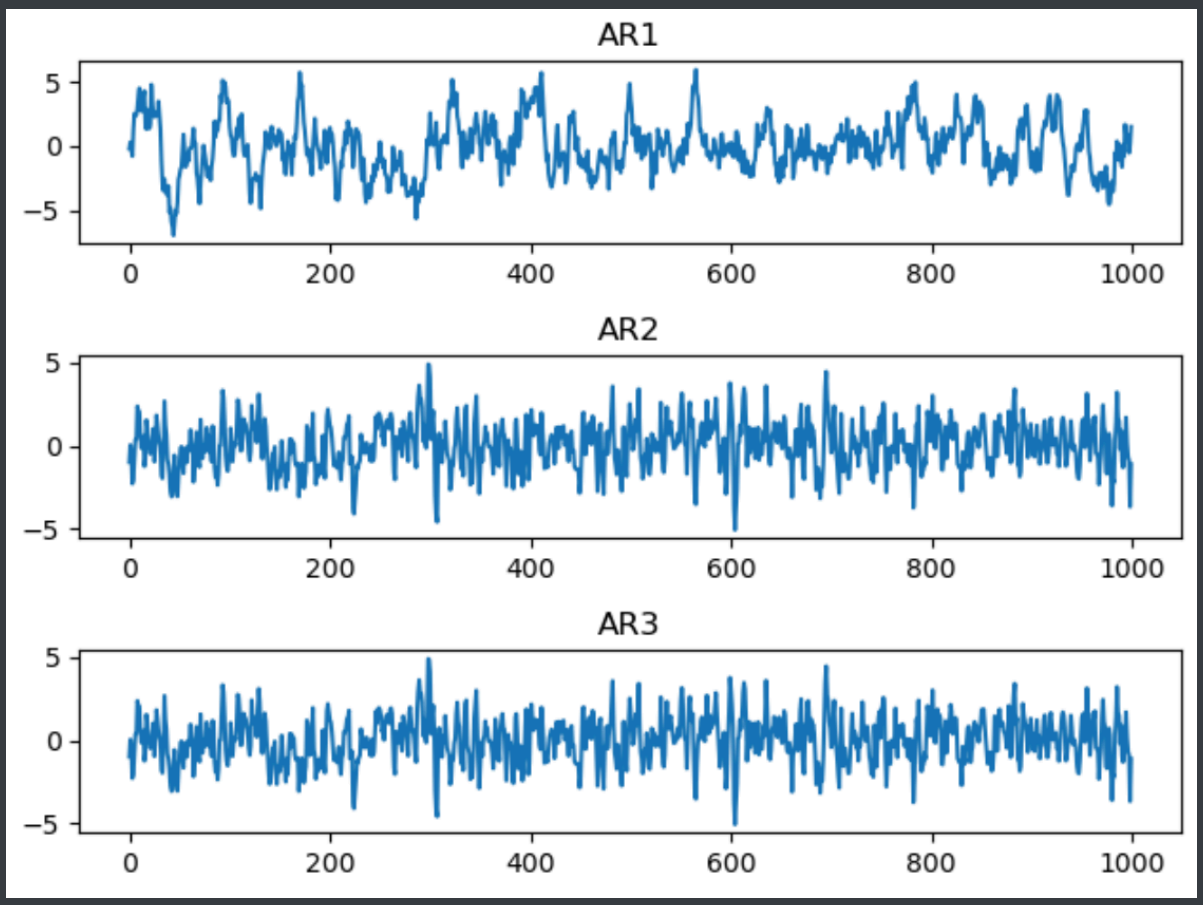
The breaking of the normality assumption will influence parameter estimation.

Problem 3

Simulate AR(1) through AR(3) and MA(1) through MA(3) processes

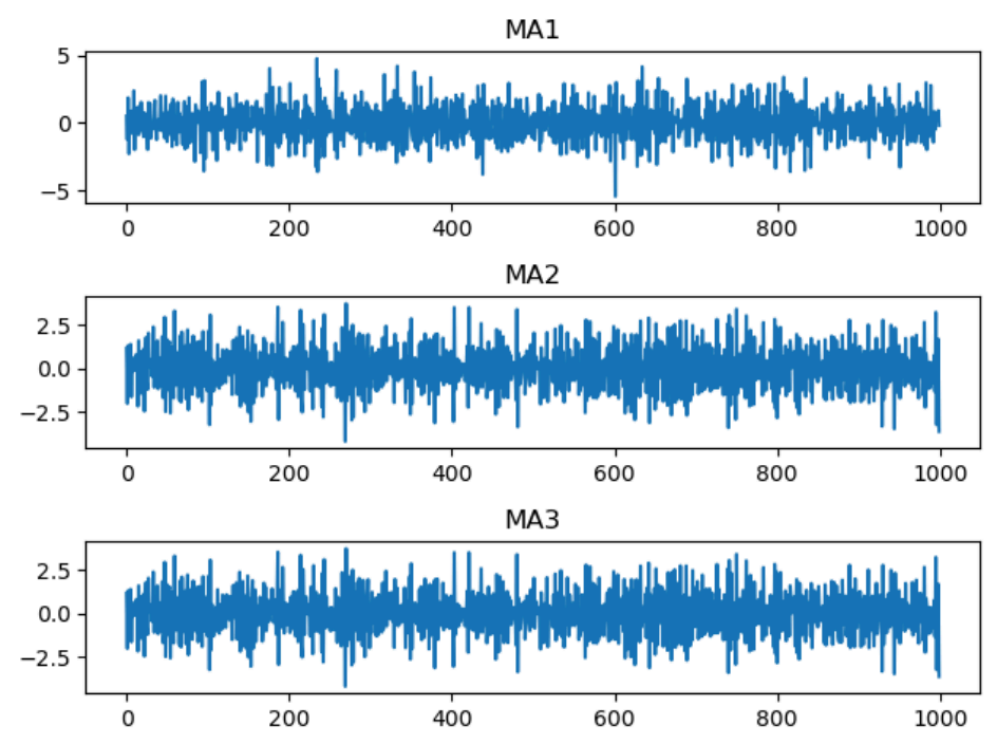
```
# AR1
ar = np.array([1, -0.9])
ma = np.array([1])
AR = ArmaProcess(ar, ma)
simulated_ar1 = AR.generate_sample(nsample=1000)

# AR2
ar = np.array([1, -0.9, 0.3])
ma = np.array([1])
AR = ArmaProcess(ar, ma)
simulated_ar2 = AR.generate_sample(nsample=1000)

# AR3
ar = np.array([1, -0.9, 0.3, 0.2])
```

```
ma = np.array([1])
AR = ArmaProcess(ar, ma)
simulated_ar3 = AR.generate_sample(nsample=1000)
```

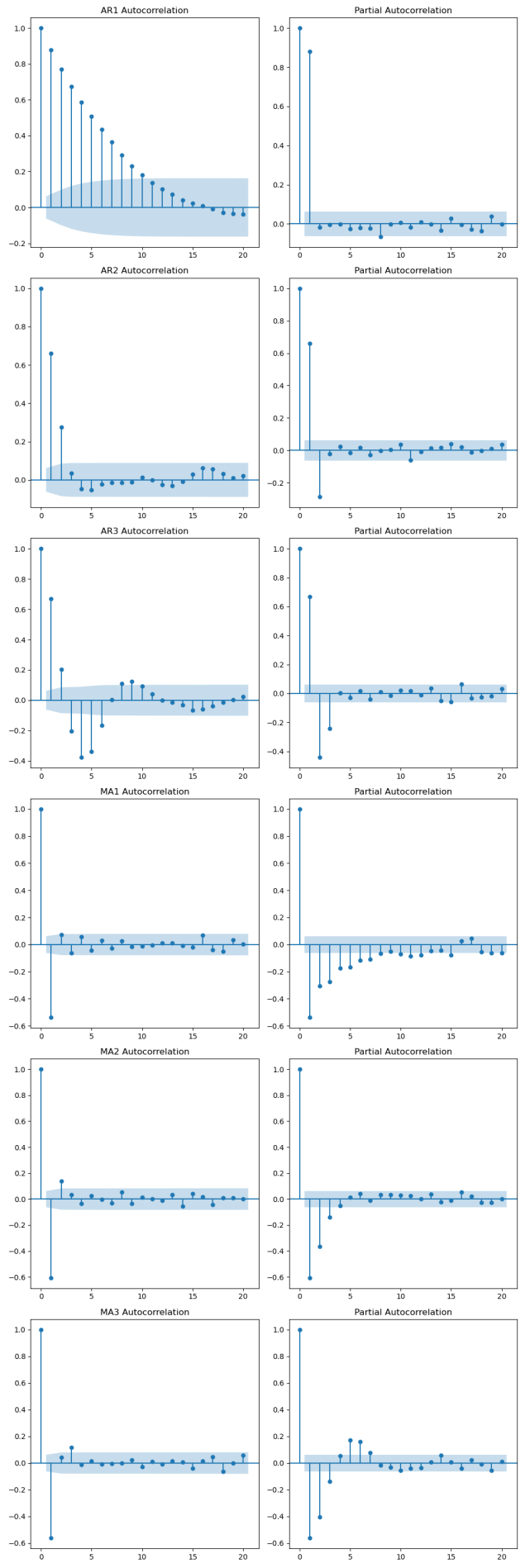



```
# MA1
ar1 = np.array([1])
ma1 = np.array([1, -0.9])
MA = ArmaProcess(ar1, ma1)
simulated_ma1 = MA.generate_sample(nsample=1000)

# MA2
ma = np.array([1, -0.9, 0.3])
ar = np.array([1])
MA = ArmaProcess(ar, ma)
simulated_ma2 = MA.generate_sample(nsample=1000)

# MA3
ma = np.array([1, -0.9, 0.3, 0.2])
```

```
ar = np.array([1])
MA = ArmaProcess(ar, ma)
simulated_ma3 = MA.generate_sample(nsample=1000)
```



ACF and PACF


AR1 Autocorrelation
Partial Autocorrelation
AR2 Autocorrelation
Partial Autocorrelation
AR3 Autocorrelation
Partial Autocorrelation
MA1 Autocorrelation
Partial Autocorrelation
MA2 Autocorrelation
Partial Autocorrelation
MA3 Autocorrelation
Partial Autocorrelation

Identify the type and order of each process

If a process is an AR process, its autocorrelation will decrease, or oscillate to decrease slowly, and the number of lags that are significantly differ from 0 in the partial autocorrelation indicates the order of this AR process.

If a process is an MA process, its partial autocorrelation will decrease, or oscillate to decrease slowly, and the number of lags that are significantly differ from 0 in the autocorrelation indicates the order of this MA process.