WEEK02

Problem 1

The values are same.
Using data for problem 1, we can find the the conditional equation when x is given, the expectation of y will be 0.04 + 0.43 * x, which is the same using OLS equation.

For conditional equation, E(y|x) = E(y) + cov(x,y) * std(x)^(-1) * (x - E(x))
= (cov(x,y) * std(x)^(-1)) * x + E(y) - cov(x,y) * std(x)^(-1) * E(x)
= beta1 * x + beta0, which is the same as OLS equation.

```python
import numpy as np
import pandas as pd
import statsmodels.api as sm

#Problem1
df1 = pd.read_csv('/Users/kunyu/Desktop/590week02.csv')
x = (df1['X']).tolist()
y = (df1['y']).tolist()

E_y = np.mean(y)
E_x = np.mean(x)
var_x = np.cov(x)

cov_xy = (np.cov(x,y))[0][1]


coefficient= cov_xy*(1/var_x)
intercept = E_y - coefficient * E_x

E_condition = "conditional distribution is %.2f + %.2f * x" %(intercept, coefficient)
print(E_condition)

X = sm.add_constant(x)
model = sm.OLS(y,X)
results = model.fit()
#print(results.params)
#print(results.summary())
E_ols= "OLS equation is %.2f + %.2f * x" %(results.params[0],results.params[1])
print(E_ols)
```

```
In [3]: runfile('/Users/kunyu/Desktop/Fintech590-Week02problem1.py', wdir='/Users/kunyu/Desktop')
conditional distribution is 0.04 + 0.43 * x
OLS equation is 0.04 + 0.43 * x
```

Problem 2

The error vector generally fits well with the assumption of normally distribution. As normally distribution has the character of skewness of 0 and kurtosis of 3. The results using data for problem 2 shows that the skewness of error vector is -0.267, and the kurtosis is 3.193.

```python
#Problem2 OLS
df2 = pd.read_csv('/Users/kunyu/Desktop/590week02problem2.csv')
x = (df2['x']).tolist()
y = (df2['y']).tolist()

X = sm.add_constant(x)
model = sm.OLS(y,X)
results = model.fit()
print(results.params)
#print(results.summary())

error = [0]*100
#print(error)
beta0 = results.params[0]
beta1 = results.params[1]

for i in range(len(x)):
    error[i] = y[i]-beta0-beta1*x[i]

print(error)
#distribution
print(stats.normaltest(error))
print(stats.shapiro(error))

mean = np.mean(error)
print(mean)
print('error mean is %.3f' % mean)

median = np.median(error)
var = np.var(error)
print('error median is %.3f' % median)
print('error variance is %.3f' % var)
skewness = stats.skew(error)
print('error skewness is %.3f' % skewness)
kurtosis = stats.kurtosis(error)
print('error kurtosis is %.3f' % kurtosis)
```

```
In [5]: runfile('/Users/kunyu/Desktop/Fintech590-Week02Problem2.py', wdir='/Users/kunyu/Desktop')
[0.1198362  0.60520482]
[-0.8384847915846739, 0.8352958594003586, 1.0274282526698084, 1.3197106992373573, -0.15231659630213926, -0.3864169595578605, 1.2847461070305262, 0.6785720966745814, -0.23279104376434817,
0.6849860543708131, 0.9047944060344617, 1.0388232599892298, 0.8818817298937349, 0.14094187587172136, 0.5944301747779469, 0.7176045519295762, 0.36758745669055026, -0.38943500370551165,
4.124036856902153, -0.05680601374039296, 0.6684267081043479, -0.9883759483843952, -1.3155729647089247, 0.2653768201451596, 0.41153462381063455, 0.7788615049781014, -1.8446537155259533,
1.0690740752232624, 1.8206886114670378, -0.9863918900701755, -0.7523942072350357, -1.0195098331622008, 0.4891546424886659, -1.6436498971209117, -0.2732363967510447, 1.1878711675980966,
0.9734158138051052, 0.13851152166001013, 0.4152964639031534, 1.1291488857209808, 0.3136963237058097, -0.7848350469020167, 0.2665901009717343, 0.5056996830456314, -1.6773841322197243,
0.6590219218180261, -0.2588123861947572, -1.9979391883638045, -0.6402635828789253, 1.52109106277079, -0.9268598826650202, -1.7115898860481118, 0.6346101097285332, 0.5039821589089635,
-0.3686530427388973, 0.08488123356319202, -1.052940042654939, -5.083892354341709, -0.5982077258513692, 1.1606906935876267, 1.6290197934642148, 0.5242746727647194, -0.042992716902360395,
0.5752575723786353, -1.4669367545127352, 1.5428134795833766, 0.2599654518438341, -1.278972586104232, 0.3044043433312187, -0.9898993658810766, 0.20064729975748652, -1.2689834773503519,
0.6849690938109821, -0.28213249696536735, -1.1177084881653894, 0.7302176390942249, -1.2016154231359915, 1.2630455085805001, 0.46058222443337754, -0.7817321847029605, 3.5316800205553722,
1.1787799053142016, 1.001982336759023, -0.5171168338263349, 0.7493718427959264, -0.8907752369849438, 0.4779263093725918, -0.67167181279711, -0.476875619385893, 0.3344297772995538,
-1.7770621428362323, -0.8541732836412241, -1.523216675392819, 0.517431086896601, -1.0716923332761124, -1.590263767391335, -1.694848149231281, 0.434877660067011, 0.4022611783382559,
-0.9223188239621974]
NormaltestResult(statistic=14.146365353030735, pvalue=0.0008475313938411151)
ShapiroResult(statistic=0.938385546207428, pvalue=0.00015389148029498756)
-1.8873791418627663e-17
error mean is -0.000
error median is 0.263
error variance is 1.436
error skewness is -0.267
error kurtosis is 3.193
```

Fitting using MLE. Compared to the assumption of normality, a T distribution go errors fits better. From the results, the maximum of likelihood function of Normality is -159.99, and for T distribution is -155.47. So the latter performs better. Breaking the normality assumption may results in a better fit.

The fitted parameters of normal distribution:
Intercept: 0.12
Beta: 0.61
Std: 1.20
AIC of normal distribution is 165

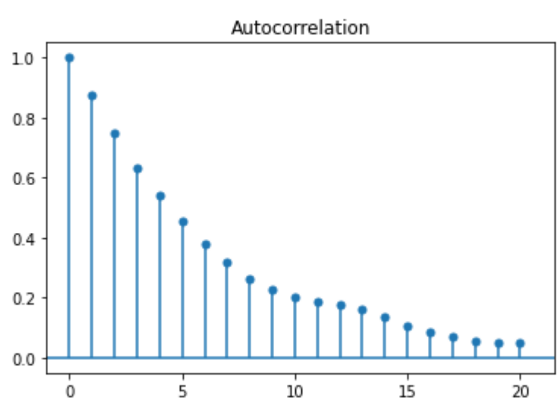
The fitted parameters of T distribution:
Intercept: 0.14
Beta: 0.56
Std: 0.97
N: 6.28
AIC of T distribution is 163

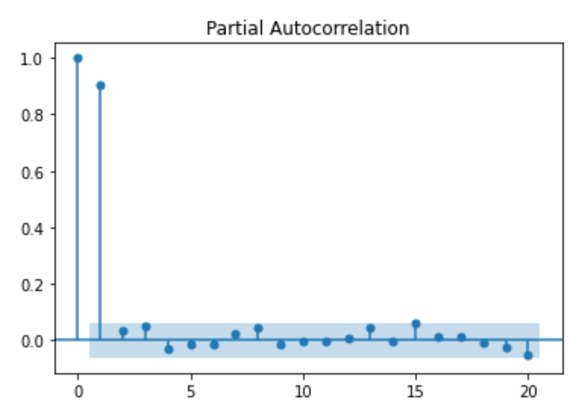
```
df = pd.read_csv('/Users/kunyu/Desktop/590week02problem2.csv')

y = df.y
x = df.x

def norm_ll(params):
    intercept, beta, std = params[0], params[1], params[2]
    #y_pred = intercept + beta*x
    e = y - intercept - beta*x
    negLL = -np.sum( stats.norm.logpdf(e, loc=0, scale=std) )
    return negLL

model_normal = optimize.minimize(norm_ll, np.array([1,1,1]), method='L-BFGS-B')
print("results of normal_mle:")
print(model_normal)


def t_ll(params):
    intercept, beta, std , n = params[0], params[1], params[2], params[3]
    e = y - intercept - beta*x
    negLL = -np.sum( stats.t.logpdf(e, df=n, loc=0, scale=std) )
    return negLL

model_t = optimize.minimize(t_ll, np.array([1,1,1,1]), method='L-BFGS-B')
print("results of t_mle:")
print(model_t)
```

```
results of normal_mle:
      fun: 159.99209668620705
 hess_inv: <3x3 LbfgsInvHessProduct with dtype=float64>
      jac: array([-1.42108547e-05, -3.55271366e-04,  1.90425454e-04])
  message: 'CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH'
     nfev: 36
      nit: 7
     njev: 9
   status: 0
  success: True
        x: array([0.11983597, 0.60519934, 1.19839549])
results of t_mle:
      fun: 155.47297041165956
 hess_inv: <4x4 LbfgsInvHessProduct with dtype=float64>
      jac: array([-1.47792889e-04, -6.28119775e-04,  4.94537742e-04, -1.70530258e-05])
  message: 'CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH'
     nfev: 90
      nit: 15
     njev: 18
   status: 0
  success: True
        x: array([0.14261189, 0.55756334, 0.97126805, 6.27649399])
```

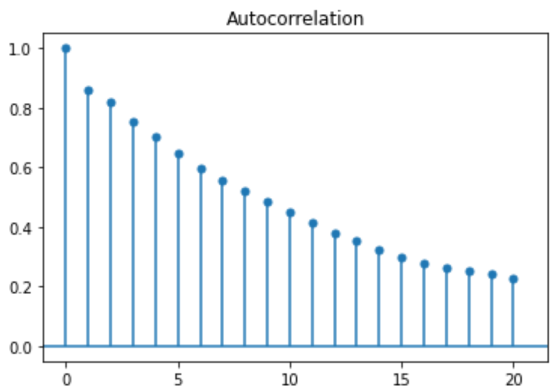
Problem 3

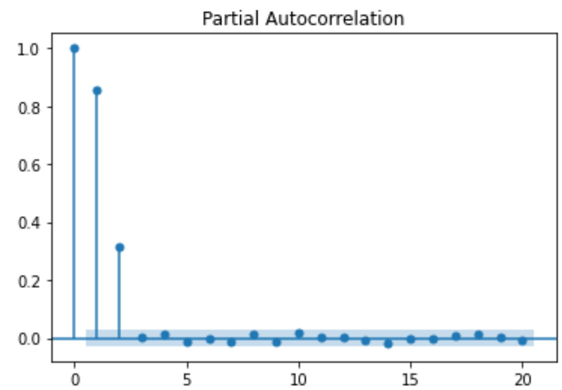
ACF and PACF of AR(1):



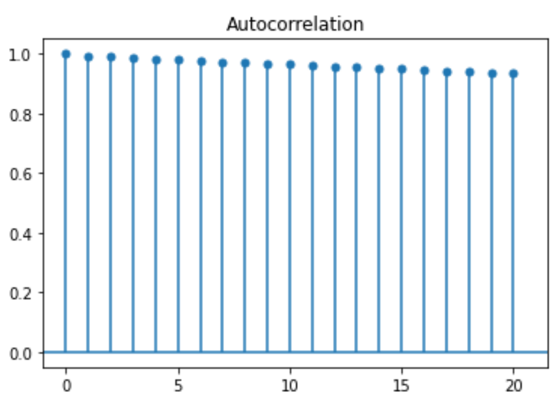


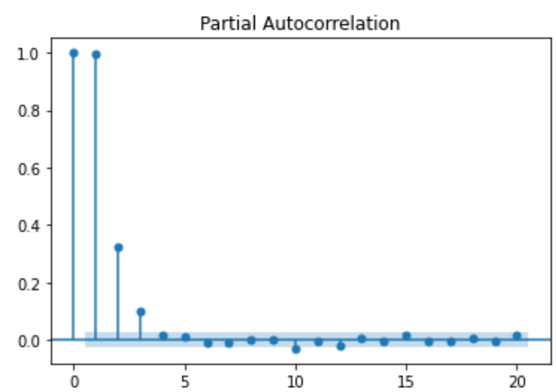
ACF and PACF of AR(2):



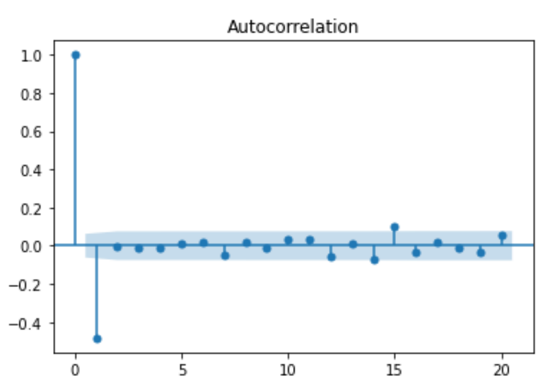


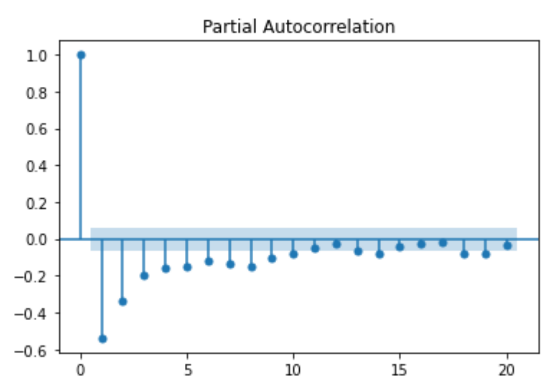
ACF and PACF of AR(3):



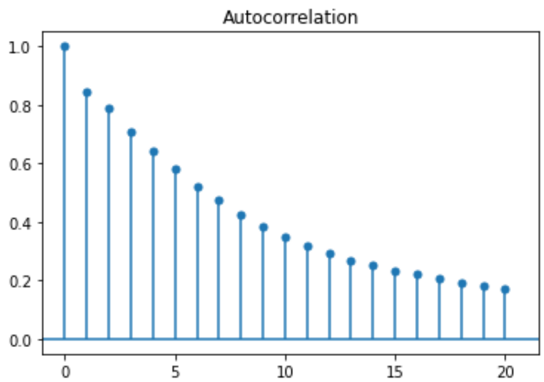


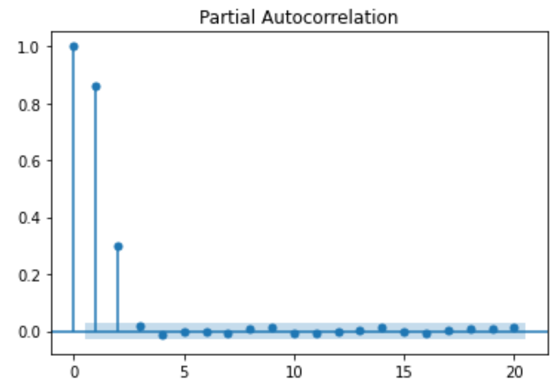
ACF and PACF of MA(1):



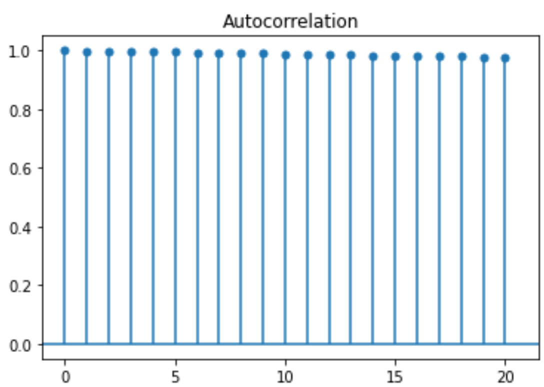


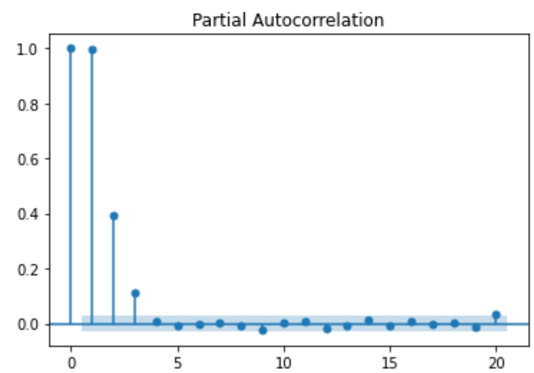
ACF and PACF of MA(2):





ACF and PACF of MA(3）：





- An AR signature corresponds to a PACF plot displaying a sharp cut-off and a more slowly decaying ACF;
- An MA signature corresponds to an ACF plot displaying a sharp cut-off and a PACF plot that decays more slowly.

Terms	ACF	PACF
AR	Geometric	p significant lags
MA	q significant lags	Geometric
ARMA	Geometric	Geometric