THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

TIME SERIES MODELING & ANALYSIS

Instructor Name: Reza Jafari

Lab#: 7

Submitted by Dinesh Kumar Padmanabhan

Date: 05-Nov-2020

ABSTRACT

In this lab continuation to our time series models we define autoregressive (AR) model and then consider a different simple time series model, the moving average (MA) model. Putting both models together to create one more general model will give us the autoregressive moving average (ARMA) model.

In an autoregressive model, we forecast the variable of interest using a linear combination of past values of the variable. On the other hand, rather than using past values of the forecast variable in a regression, a moving average model uses past forecast errors in a regression-like model.

INTRODUCTION

Auto- Regressive (AR) Models

In a multiple regression model, we forecast the variable of interest using linear combination of predictors. In an autoregressive model, we forecast the variable of interest using a linear combination of past values of the variable. An AR process of order na AR(na) can be written as :

$$y(t) + a_1 y(t-1) + a_2 y(t-2) + \ldots + a_{n_a} y(t-n_a) = \epsilon(t)$$

where y(t) is the variable of interest and e(t) is white noise (WN ~ (0, σ2 e).

This is like a multiple regression but with lagged values of y(t) as predictors. The term autoregressive indicates that it is a regression of variable against itself.

Moving Average (MA) Models

Rather than using past values of the forecast variable in a regression, a moving average model uses past forecast errors in a regression-like model:

$$y(t) = \epsilon(t) + b_1 \epsilon(t-1) + b_2 \epsilon(t-2) + \ldots + b_{n_b} \epsilon(t-n_b)$$

(t) is white noise WN ~ (0, σ2). This is called a moving average model of order nb , MA(nb).

In MA model, each y(t) can be thought of as weighted moving average of the past few forecast errors.

Like the AR models, the variance of the σ 2 e will only change the scale of series , not the patterns.

METHOD, THEORY & PROCEDURES

Method:

1. Programming Language: Python

Libraries used: Some basic libraries used for analysis & model building are mentioned below

library(Numpy) - large collection of high-level mathematical functions to operate on these arrays.

library (Pandas) – For Data manipulation and analysis

library(Matplotlib) – is a system for declaratively creating graphics

library(Math) –To Compute mathematical calculations

library (statsmodels) – Import statistical models

library (scipy) – Scientific Computations

Theory:

We want to eliminate features and find the best multiple linear regression model. To Plot the forecast accuracy of above-mentioned methods for the given data set and determine which method performs better.

Procedure:

I shall be looking at the results of auto-regressive and moving average methods of time series models. Perform various plots and infer about it in my analysis. And through my exploration I shall try to identify which methods perform better and draw inferences.

The Dataset will be explored in following stages:

1. **Data Exploration (EDA)** – looking at the models and making inferences about the data.
2. **Data Visualization** – Plotting different time series plots for the regression method and forecast accuracy.
3. **Testing** – Running Autocorrelation, Pearson correlation test to identify the correlation between errors.

ANSWERS TO QUESTIONS

File - unknown

```
C:\ProgramData\Anaconda3\python.exe "C:\Program Files\
JetBrains\PyCharm 2019.3.1\plugins\python\helpers\pydev\
pydevconsole.py" --mode=client --port=51347

import sys; print('Python %s on %s' % (sys.version, sys.
platform))
sys.path.extend(['C:\\Users\\nsree_000\\Desktop\\Python-
Quiz', 'C:/Users/nsree_000/Desktop/Python-Quiz'])

Python 3.7.4 (default, Aug  9 2019, 18:34:13) [MSC v.1915
64 bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more
information
IPython 7.8.0 -- An enhanced Interactive Python. Type '?'
for help.
PyDev console: using IPython 7.8.0

Python 3.7.4 (default, Aug  9 2019, 18:34:13) [MSC v.1915
64 bit (AMD64)] on win32
In[2]: runfile('C:/Users/nsree_000/Desktop/Python-Quiz/TIME
 SERIES/Lab7_1.py', wdir='C:/Users/nsree_000/Desktop/Python
-Quiz/TIME SERIES')
****************************** AR PROCESS
 *********************************
The mean of AR(2) is  3.2530814861632527 The variance of
the AR(2) is  3.079187081046015
First five values of y(t) are [2.88314769 3.45313107 1.
11766902 2.23760417 3.22103765]
ADF Statistic: -10.658356
p-value: 0.000000
Critical Values:
    1%: -3.437
    5%: -2.864
    10%: -2.568
Experimental mean of y(t) is = 3.2530814861632527
Experimental variance of y(t) is = 3.079187081046015
First five values of y(t) using dlsim command are [2.
883147693612942, 3.453131073281014, 1.1176690229639972, 2.
2376041718643847, 3.2210376515912045]
Enter the number of samples: >? 1000
Enter the order # of the AR process: >? 2
You need to enter 2 coefficients since the order of the AR
process is 2
Enter the coefficient 1 of AR process :>? 0.5
Enter the coefficient 2 of AR process :>? 0.2
Actual Coefficients for AR(2) with 1000 samples are [0.5, 0
```

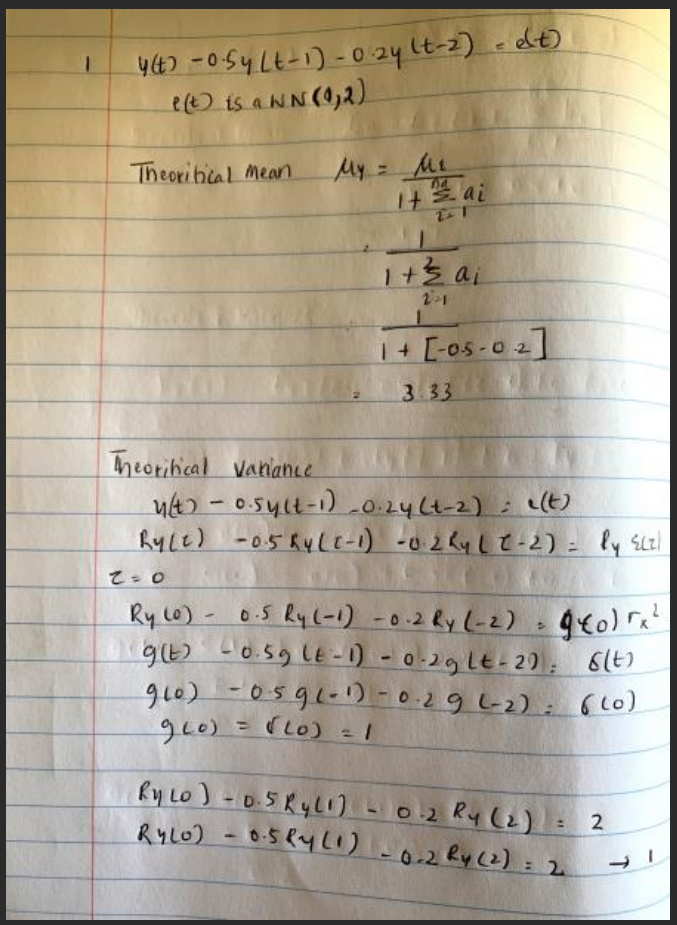
Page 1 of 2

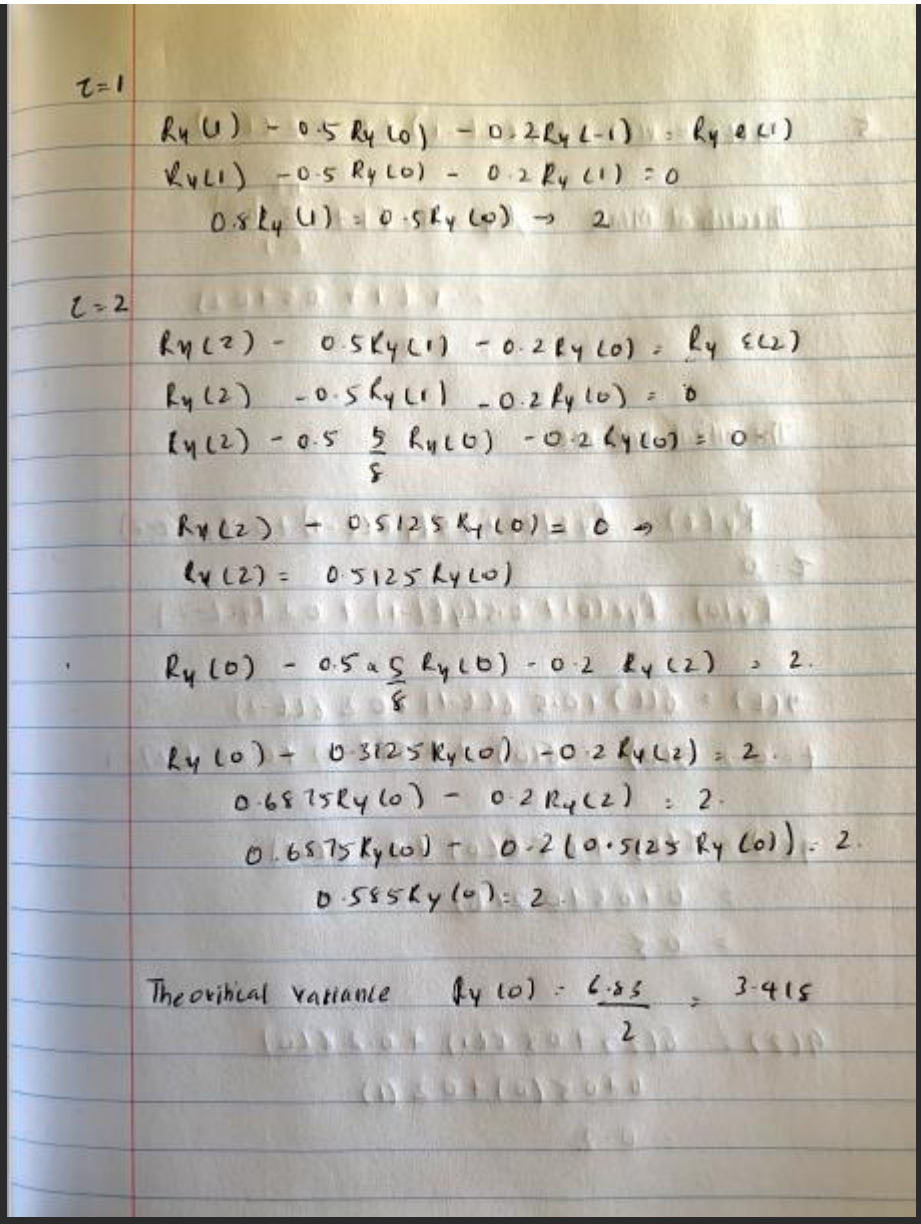
File - unknown

```
30 .2]
31 Estimated Coefficients for AR(2) with 1000 samples are [0.
   218, -0.108]
32 Actual Coefficients for AR(2) with 1000 samples are [-0.5
   , -0.2]
33 Estimated Coefficients for AR(2) with 1000 samples are [-0.
   613, -0.332]
34 Actual Coefficients for AR(2) with 5000 samples are [-0.5
   , -0.2]
35 Estimated Coefficients for AR(2) with 5000 samples are [-0.
   618, -0.318]
36 Actual Coefficients for AR(2) with 10000 samples are [-0.5
   , -0.2]
37 Estimated Coefficients for AR(2) with 10000 samples are [-0
   .623, -0.318]
38 ***************************** MA PROCESS
    **********************************
39 The mean of MA(2) is  1.6635778510418626 The variance of
   the MA(2) is  2.308074254578314
40 First five values of y(t) are [2.88314769 3.45313107 0.
   3968821  0.79769186 2.13566827]
41 ADF Statistic: -15.873015
42 p-value: 0.000000
43 Critical Values:
44     1%: -3.437
45     5%: -2.864
46     10%: -2.568
47 Experimental mean of y(t) is = 1.6635778510418626
48 Experimental variance of y(t) is = 2.308074254578314
49 First five values of y(t) using dlsim command are [2.
   883147693612942, 3.453131073281014, 0.3968820995607616, 0.
   7976918648215425, 2.1356682734494843]
50
```

Page 2 of 2

```
#%%=========================================================================
# 1: Using the Python program and appropriate libraries perform the following
tasks:
# Let consider an AR(2) process as y(t) – 0.5y(t – 1) – 0.2y(t – 2) = e(t)
# Where e(t) is a WN (1,2).
# %%-------------------------------------------------------------------------
# a. Find the theoretical mean and variance of y(t). (no need to use python).
# b. Using python, create a for loop that simulates above process for 1000
samples. Assume all initial conditions to be zero.
# c. Using the generated samples in part b and numpy package, find the
experimental mean and variance. Compare your answer with part a.
Write down your observations.
# d. Plot the y(t) with respect to number of samples.
# e. Using the python code, developed in previous labs, calculate autocorrelations
for 20 lags and plot them versus number of lags. Write down your observation about
the ACF of above process.
# f. Display the first 5 values of y(t) at the console.
# g. Apply the ADF-test and check if this is a stationary process. Explain your
answer
# %%-------------------------------------------------------------------------

a.
```

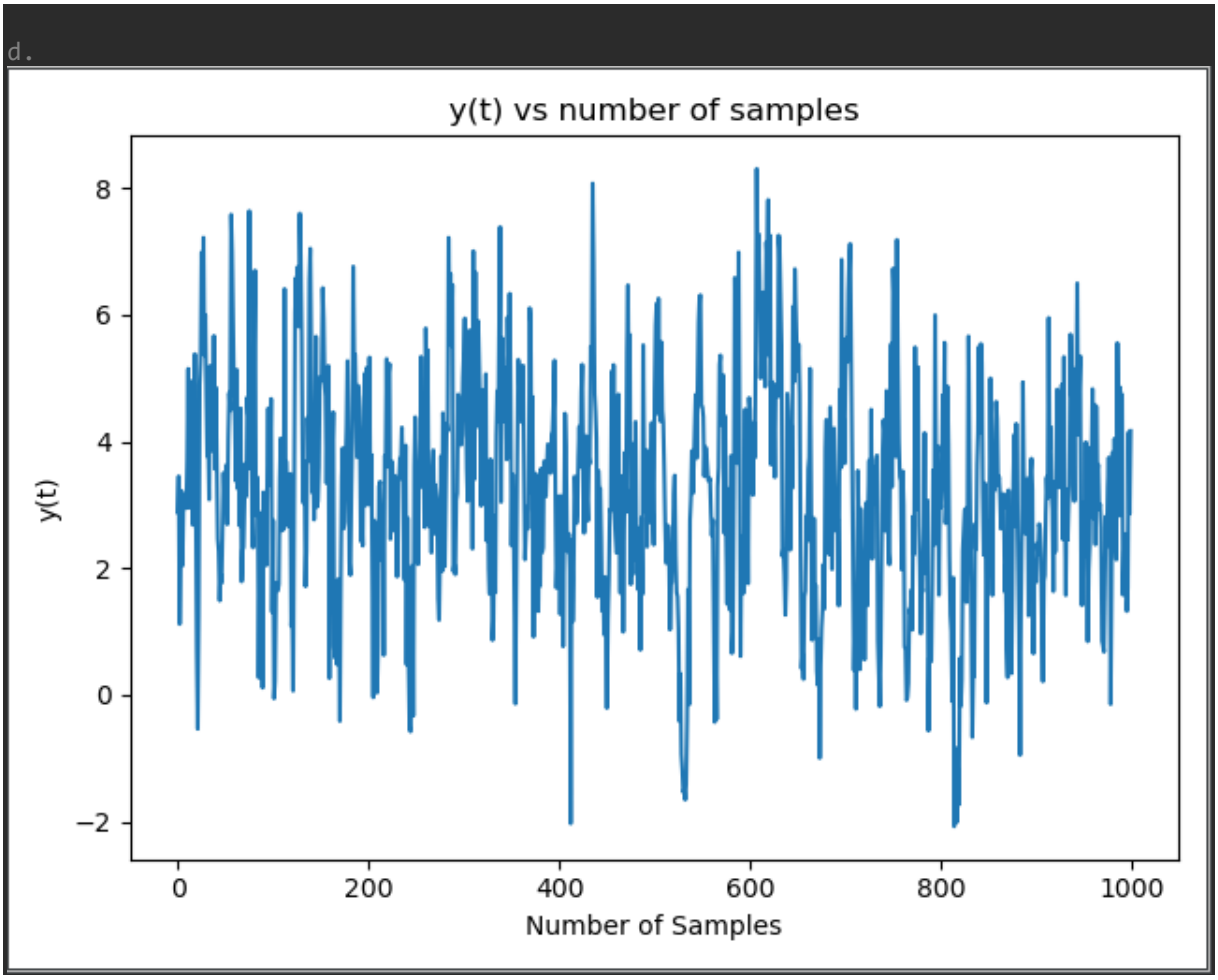




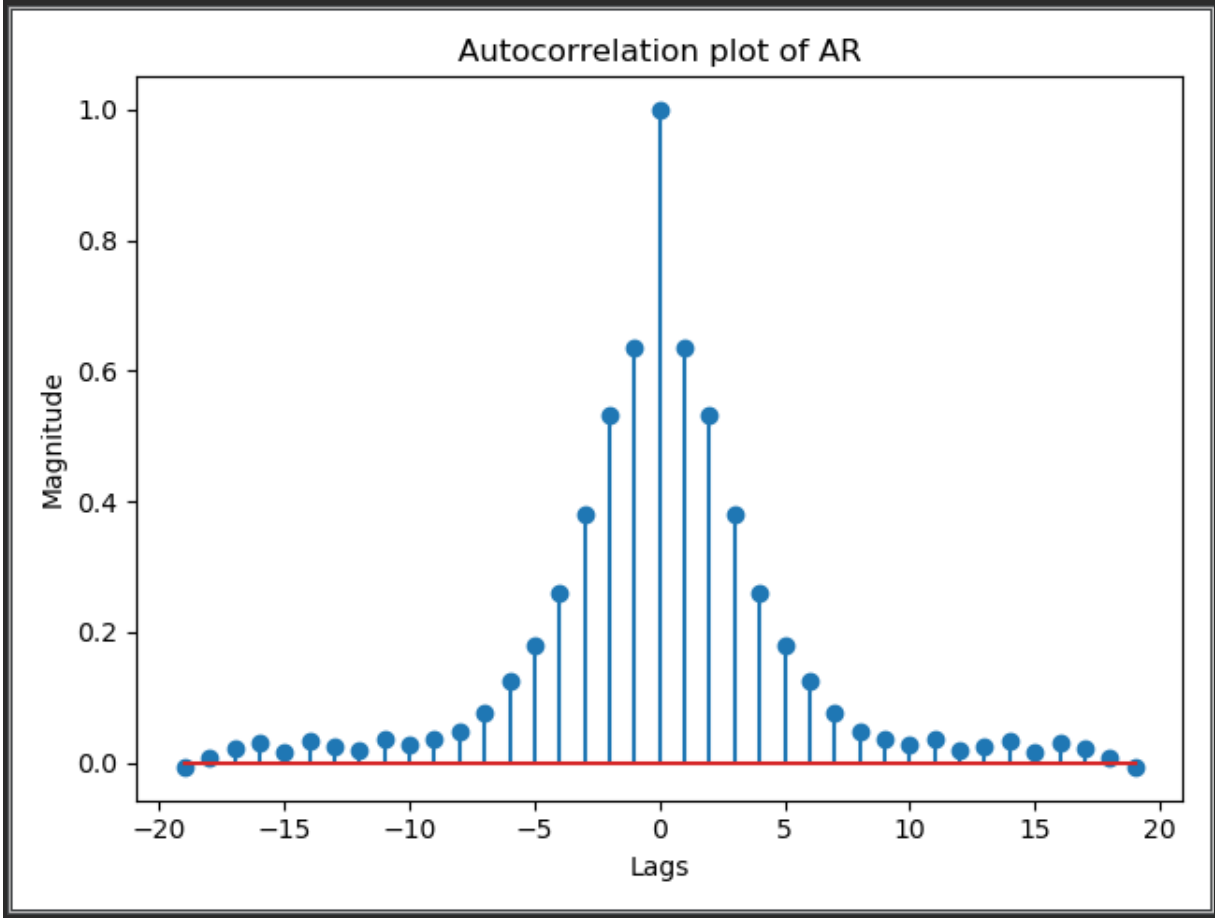
c.
The mean of AR(2) is 3.2530814861632527
The variance of the AR(2) is 3.079187081046015

From a and c we can see that theoretical and simulated values matches.

d.


y(t) vs number of samples
y(t)
Number of Samples
8
6
4
2
0
−2
0
200
400
600
800
1000

e.



Auto correlation of white noise have a strong peak at 0 and tending to zero for all other lags which is basically an impulse.

Histogram as an gaussian distribution.

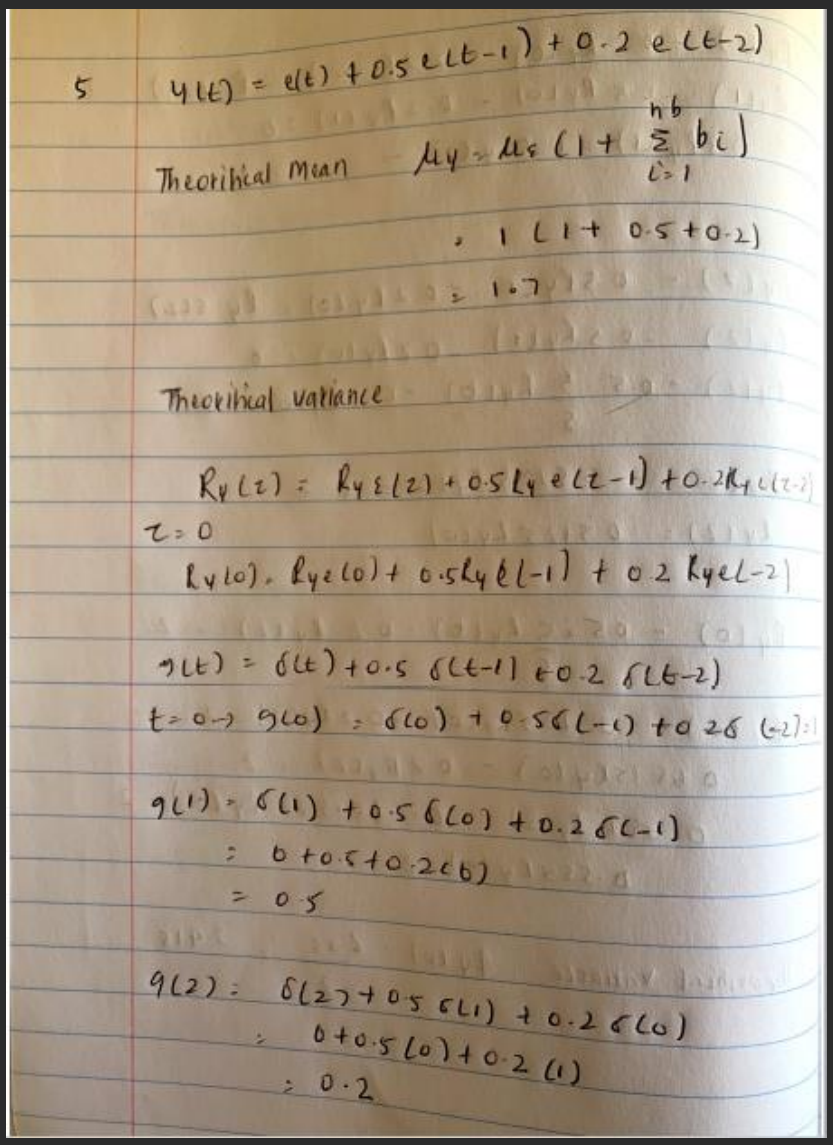
From the time plot we can see the variables are independent and identically distributed with a mean of zero. This means that all variables have the same variance (sigma^2) and each value has a zero correlation with all other values in the series.

```
f.

First five values of y(t) are [2.88314769 3.45313107 1.11766902 2.23760417
3.22103765]

g.

#### ADF test on Samples ####
ADF Statistic: -10.658356
p-value: 0.000000
Critical values :
        1%: -3.437
        5%: -2.864
        10%: -2.568

The p-value is less than 5%, we strong evidence against the null hypothesis,
so we reject the null hypothesis.
As per the test results above, the p-value is 0.0 which is < 0.05 therefore we
reject the null in favour of the alternative hypothesis that the time series is
stationary.
```

```
#%%=============================================================================
#2. Using the “scipy” python package and “dlsim” command, simulate the AR(2)
process in question
# a. Display the first 5 values of y(t) at the console.
# b. Show that your answer to the previous part is identical to the answer in part
d of previous question.
# %%-----------------------------------------------------------------------------

Experimental mean of y(t) is = 3.2530814861632527

Experimental variance of y(t) is = 3.079187081046015

First five values of y(t) using dlsim command are
[2.883147693612942, 3.453131073281014, 1.1176690229639972, 2.2376041718643847,
3.2210376515912045]

We can see that the answers match with previous question even though they are
different methods

#%%=============================================================================
#3. Write the AR(2) process in question 1, as multiple regression model and using
the least square estimate (LSE),
# estimate the true parameters 𝑎1 and 𝑎2 (0.5, 0.2).
# Display the estimated parameters values at the console.
# What is the effect of additional samples on the accuracy of the estimate?
# Justify the answer by running your code for 5000 and 10000 data samples.
# %%-----------------------------------------------------------------------------

Actual Coefficients for AR(2) with 1000 samples are [0.5, 0.2]
Estimated Coefficients for AR(2) with 1000 samples are [0.506, 0.179]

Actual Coefficients for AR(2) with 1000 samples are [-0.5, -0.2]
Estimated Coefficients for AR(2) with 1000 samples are [-0.497, -0.215]

The figure implies that as we keep increasing the sample size to infinity, the AR
estimate converges to the true value with probability 1. We can see that our
estimated coefficients are close to the actual values.

Actual Coefficients for AR(2) with 5000 samples are [-0.5, -0.2]
Estimated Coefficients for AR(2) with 5000 samples are [-0.495, -0.195]

Actual Coefficients for AR(2) with 10000 samples are [-0.5, -0.2]
Estimated Coefficients for AR(2) with 10000 samples are [-0.499, -0.194]
```

```
#%%=========================================================================
#4. Generalized your code in the previous question, such when the code runs it
asks a user the
# following questions:
# a. Enter number of samples :
# b. Enter the order # of the AR process:
# c. Enter the corresponding parameters of AR process :
    # Your code should simulate the AR process based on the entered information
(a, b, c) and estimate the AR
    # parameters accordingly. The estimated parameters must be close to the
entered numbers in part c.
    # Display the estimated parameters and the true values at the console.
# d. Increase the number of samples to 5000 and display the estimated parameters.
# e. Increase the number of samples to 10000 and display the estimated parameters.
# f. Write down your observation on the effect of the additional samples on the
accuracy of the estimation.
# %%-------------------------------------------------------------------------

Actual Coefficients for AR(2) with 1000 samples are [0.5, 0.2]
Estimated Coefficients for AR(2) with 1000 samples are [0.506, 0.179]

Actual Coefficients for AR(2) with 1000 samples are [-0.5, -0.2]
Estimated Coefficients for AR(2) with 1000 samples are [-0.497, -0.215]

Actual Coefficients for AR(2) with 5000 samples are [-0.5, -0.2]
Estimated Coefficients for AR(2) with 5000 samples are [-0.495, -0.195]

Actual Coefficients for AR(2) with 10000 samples are [-0.5, -0.2]
Estimated Coefficients for AR(2) with 10000 samples are [-0.499, -0.194]

The figure implies that as we keep increasing the sample size to infinity, the AR
estimate converges to the true value with probability 1. We can see that our
estimated coefficients are close to the actual values.
```

```
#%%=========================================================================
#5. Let consider an MA(2) process as y(t) = e(t) + 0.5e(t – 1) + 0.2e(t – 2) Where
e(t) is a WN (1,2).
# a. Find the theoretical mean and variance of y(t). (no need to use python).
# b. Using python, create a for loop simulate above process for 1000 samples.
Assume all initial conditions to be zero.
# c. Plot the y(t) with respect to number of samples.
# d. Using the python code, developed in previous labs, calculate autocorrelations
for 20 lags and plot them versus number of lags. Write down your observation about
the ACF of above process.
# e. Increase the data samples to 10000 and then 100000. Observe the ACF. Write
down your observation. There is a difference between the ACF of an AR process and
MA process.
# What is the main difference?
# f. Display the first 5 values of y(t) at the console.
# g. Apply the ADF-test and check if this is a stationary process. Explain your
answer.
# %%-------------------------------------------------------------------------

a.
```



$R_{ye}(0) = g(0)\sigma_x^2 = 1 \times 2 = 2$.

$R_{ye}(-1) = g(1)\sigma_x^2 = 0.5 \times 2 = 1$

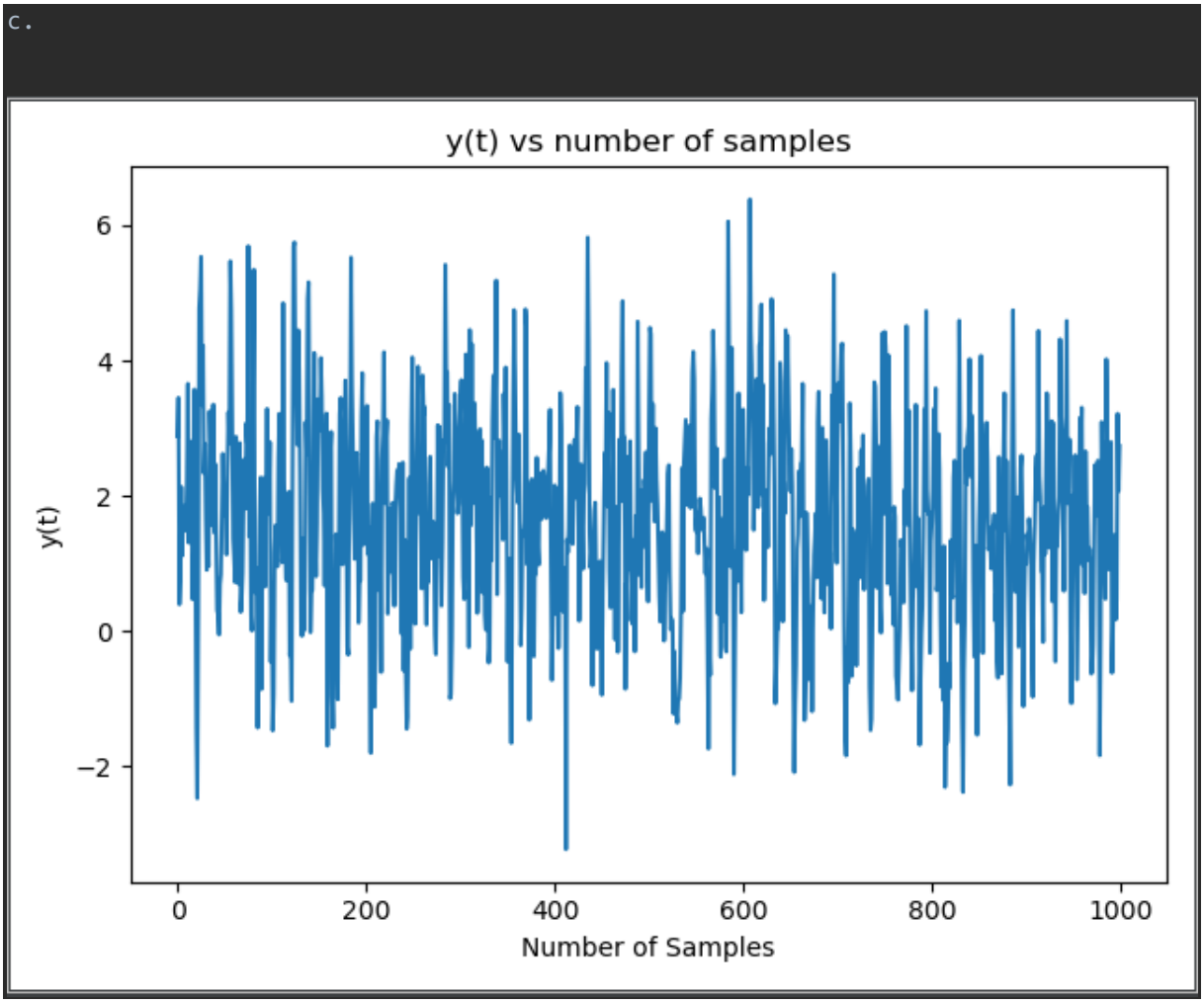
$R_{ye}(-2) = g(2)\sigma_x^2 = 0.2 \times 2 = 0.4$

$R_y(0) = R_{ye}(0) + 0.5 R_y(-1) + 0.2 R_{ye}(-2)$

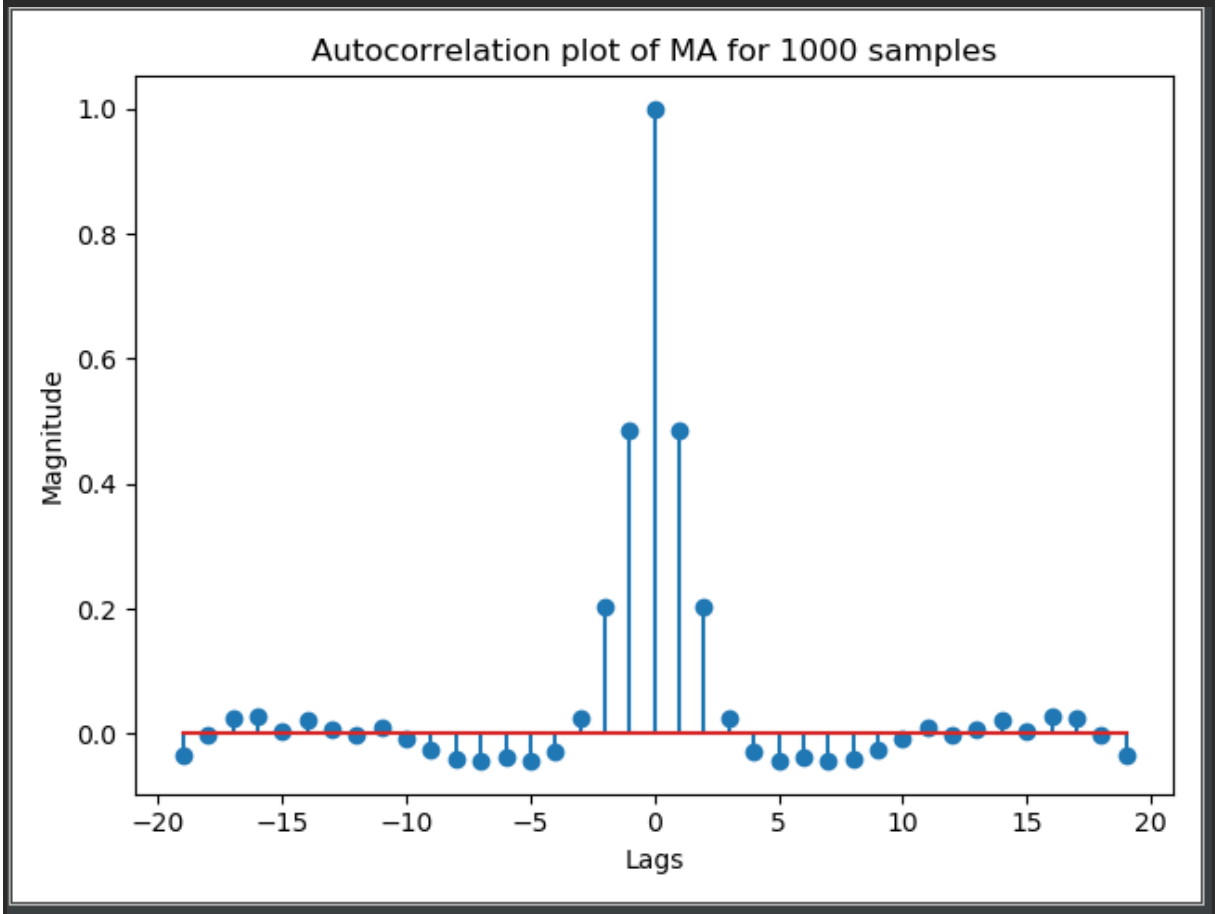
$R_y(0) = 2 + 0.5(1) + 0.2(0.4)$

Theoritical variance $R_y(0) = 2.58$.

c.

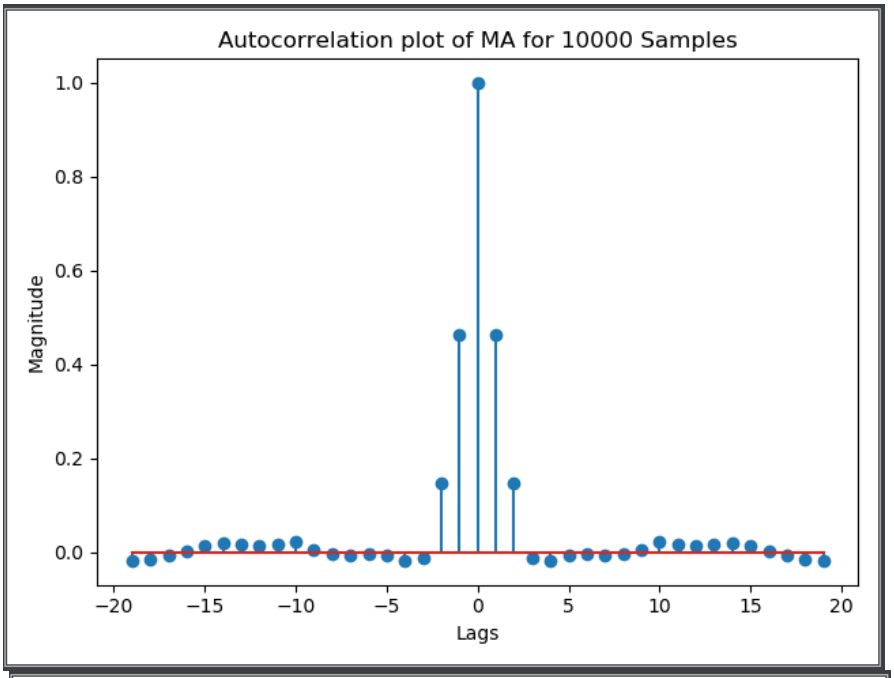

y(t) vs number of samples
y(t)
6
4
2
0
−2
0
200
400
600
800
1000
Number of Samples

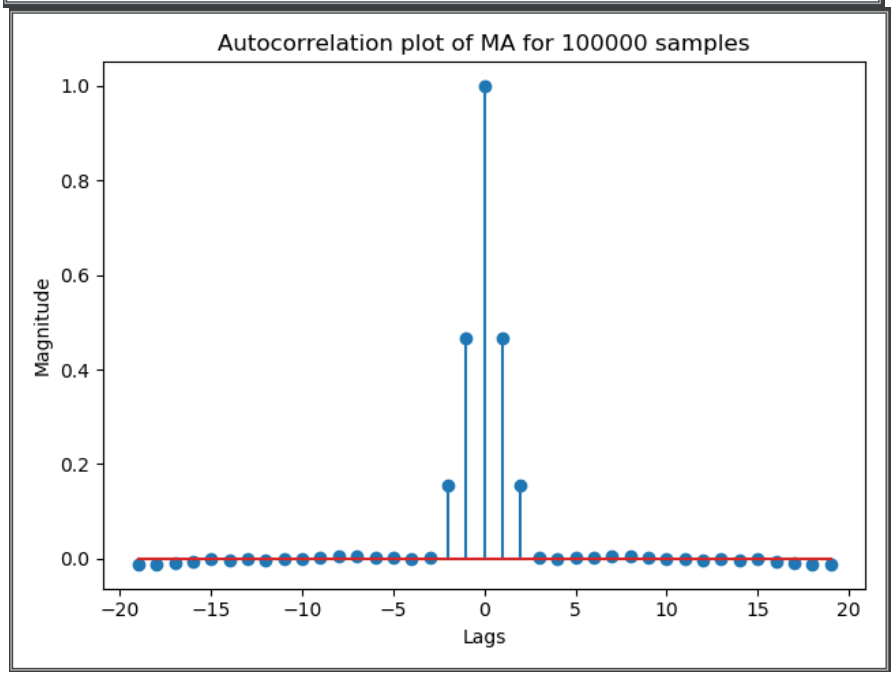
d.



Auto correlation of white noise have a strong peak at 0 and tending to zero for all other lags which is basically an impulse.
Histogram as an gaussian distribution.
From the time plot we can see the variables are independent and identically distributed with a mean of zero. This means that all variables have the same variance (sigma^2) and each value has a zero correlation with all other values in the series.

```
e.

For the AR process, we expect that the ACF plot will gradually decrease.
To define a MA process, we expect the opposite from the ACF, meaning that:
the ACF shows a sharp drop after 2 number of lags.
```





```
f.

First five values of y(t) are [ 1.3315865   1.38107223 -0.9214435  -0.6380282
0.30806399]
g.

ADF Statistic: -15.873015
p-value: 0.000000
Critical values :
        1%: -3.437
        5%: -2.864
        10%: -2.568

The p-value is less than 5%, we strong evidence against the null hypothesis,
so we reject the null hypothesis.
As per the test results above, the p-value is 0.0 which is < 0.05 therefore we
reject the null in favour of the alternative hypothesis that the time series is
stationary.

#%%=============================================================================
#6. Using the “scipy” python package and “dlsim” command, simulate the MA(2)
process in question
    # a. Display the first 5 values of y(t) at the console.
    # b. Show that your answer to the previous part is identical to the answer in
part f of the previous question.
# %%-----------------------------------------------------------------------------

Experimental mean of y(t) is = 1.6635778510418626
Experimental variance of y(t) is = 2.308074254578314

We can see that the answers match with previous question even though they are
different methods
```

CONCLUSION

An ARMA model requires the data to be stationary. A stationary series has a constant mean and a constant variance over time. For the white noise, AR and MA processes we've defined above, this requirement holds and we tested the stationarity with a Dicker Fuller test. We've discussed the definition of AR and MA models in this post as well as the ACF. We've also conclude that these kind of models can only work with stationary data or data with a trend and that they are not suitable for long term forecasting.

CHALLENGE

Calculations was little tricky to understand in the beginning, after lot of clarifications it provided clarity.

APPENDIX

```
import numpy as np
import pandas as pd
from Autocorrelation import cal_auto_corr
import matplotlib.pyplot as plt
from statsmodels.tsa.stattools import adfuller
from scipy import signal
import random
import warnings
warnings.filterwarnings("ignore")

def ADF_Cal(x):
    result = adfuller(x)
    print('ADF Statistic: %f' %result[0])
    print('p-value: %f' %result[1])
    print('Critical Values:')
    for key, value in result[4].items():
        print('\t%s: %.3f' % (key, value))
print(20 * "-" + "ARPROCESS" + 20 * "-")
T = 1000
mean = 0
std = np.sqrt(1)
np.random.seed(10)
e = np.random.normal(mean, std, size=T)
# print('The mean of WN is ', np.mean(e), 'The variance of the WN is ', np.var(e))
y = np.zeros(len(e))
for t in range(len(e)):
    if t == 0:
        y[t] = e[t]
    elif t == 1:
        y[t] = 0.5*y[t-1] + e[t]
    else:
        y[t] = 0.5*y[t-1] + 0.2*y[t-2] + e[t]

print('The mean of AR(2) is ', np.mean(y), 'The variance of the AR(2) is ',
np.var(y))

plt.figure()
plt.plot(y)
plt.xlabel('Number of Samples')
plt.ylabel('y(t)')
plt.title('y(t) vs number of samples')
plt.show()

k = 20
result = cal_auto_corr(y, k)
plt.figure()
plt.stem(range(-(k-1),k), result, use_line_collection=True)
plt.xlabel('Lags')
```

```
plt.ylabel('Magnitude')
plt.title('Autocorrelation plot of AR')
plt.show()

print('First five values of y(t) are {}'.format(y[0:5]))
ADF_Cal(y)

np.random.seed(10)
e = np.random.normal(mean, std, size=T)
num = [1,0,0]
den = [1, -0.5, -0.2]
sys = (num, den, 1)
_, y =signal.dlsim(sys, e)
y = [item for sublist in y for item in sublist]
print('Experimental mean of y(t) is =', np.mean(y))
print('Experimental variance of y(t) is =', np.var(y))
print('First five values of y(t) using dlsim command are {}'.format(y[0:5]))

plt.figure()
plt.plot(y)
plt.xlabel('Number of Samples')
plt.ylabel('Y(t)')
plt.title('y(t) vs number of samples')
plt.show()

def AR(T, order, params):
    mean = 1
    std = np.sqrt(2)
    np.random.seed(10)
    e = np.random.normal(mean, std, size=T)
    # print('The mean of WN is ', np.mean(e), 'The variance of the WN is ',
np.var(e))
    num = [1, 0, 0]
    den = [1]
    den.extend(params)
    sys = (num, den, 1)
    _, y = signal.dlsim(sys, e)
    y = [item for sublist in y for item in sublist]
    T_new = len(y) - order - 1
    Y = pd.DataFrame(y[order:len(y)])
    X = np.zeros((T_new + 1, order))
    y_l = list(y)
    k = 1
    for j in range(order):
        for i in range(T_new + 1):
            X[i][j] = y_l[order+i-k]
        k += 1
    X = -1 * pd.DataFrame(X)
    x_transpose = X.transpose()
    coeff = np.linalg.inv(np.array(x_transpose.dot(X))).dot(x_transpose.dot(Y))
    coeff = [round(item, 3) for sublist in coeff for item in sublist]
    print('Actual Coefficients for AR({}) with {} samples are {}'.format(order, T,
params))
    print('Estimated Coefficients for AR({}) with {} samples are {}'.format(order,
T, coeff))
    return coeff, X, Y

T = int(input('Enter the number of samples: '))
```

```
order = int(input('Enter the order # of the AR process: '))
print("You need to enter {} coefficients since the order of the AR process is
{}".format(order, order))
params = []
for i in range(1, order+1):
    params.append(float(input('Enter the coefficient {} of AR process
:'.format(i))))

coeff, X, Y = AR(T, order, params)
coeff, X, Y = AR(T=1000, order=2, params=[-0.5, -0.2])
coeff, X, Y = AR(T=5000, order=2, params=[-0.5, -0.2])
coeff, X, Y = AR(T=10000, order=2, params=[-0.5, -0.2])


print(20 * "-" + "MA PROCESS" + 20 * "-")
def MA(T):
    mean = 0
    std = np.sqrt(1)
    np.random.seed(10)
    e = np.random.normal(mean, std, size=T)
    # print('The mean of WN is ', np.mean(e), 'The variance of the WN is ',
np.var(e))
    y = np.zeros(len(e))
    for t in range(len(e)):
        if t == 0:
            y[t] = e[t]
        elif t == 1:
            y[t] = e[t] + 0.5*e[t-1]
        else:
            y[t] = e[t] + 0.5*e[t-1] + 0.2*e[t-2]
    return y

y = MA(T=1000)
print('The mean of MA(2) is ', np.mean(y), 'The variance of the MA(2) is ',
np.var(y))

plt.figure()
plt.plot(y)
plt.xlabel('Number of Samples')
plt.ylabel('y(t)')
plt.title('y(t) vs number of samples')
plt.show()

k = 20
result = cal_auto_corr(y, k)
plt.figure()
plt.stem(range(-(k-1),k), result, use_line_collection=True)
plt.xlabel('Lags')
plt.ylabel('Magnitude')
plt.title('Autocorrelation plot of MA for 1000 samples')
plt.show()

y_10000 = MA(T=10000)
y_100000 = MA(T=100000)

result = cal_auto_corr(y_10000, k)
plt.figure()
plt.stem(range(-(k-1),k), result, use_line_collection=True)
```

```
plt.xlabel('Lags')
plt.ylabel('Magnitude')
plt.title('Autocorrelation plot of MA for 10000 Samples')
plt.show()

result = cal_auto_corr(y_100000, k)
plt.figure()
plt.stem(range(-(k-1),k), result, use_line_collection=True)
plt.xlabel('Lags')
plt.ylabel('Magnitude')
plt.title('Autocorrelation plot of MA for 100000 samples')
plt.show()

print('First five values of y(t) are {}'.format(y[0:5]))
ADF_Cal(y)

T = 1000
np.random.seed(10)
e = np.random.normal(mean, std, size=T)
num = [1, 0.5, 0.2]
den = [1, 0, 0]
sys = (num, den, 1)
_, y =signal.dlsim(sys, e)
y = [item for sublist in y for item in sublist]
print('Experimental mean of y(t) is =', np.mean(y))
print('Experimental variance of y(t) is =', np.var(y))
print('First five values of y(t) using dlsim command are {}'.format(y[0:5]))

plt.figure()
plt.plot(y)
plt.xlabel('Number of Samples')
plt.ylabel('y(t)')
plt.title('y(t) vs number of samples')
plt.show()
```

REFERENCES

https://otexts.com/fpp2/#