**DATS 6313 – Time Series Analysis & Modeling**

Instructor: Reza Jafari

**Lab #5**

Bradley Reardon

2/23/2022

**Abstract:**

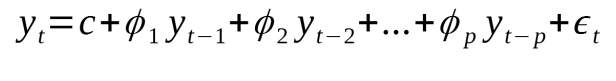
This lab pertains to implementing moving average (MA) and auto regressive (AR) models.

**Introduction:**

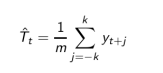
This experiment was performed to increase understanding of the application moving average (MA) and auto regressive (AR) models while creating programs to function for us.

**Method, Theory, and Procedures:**

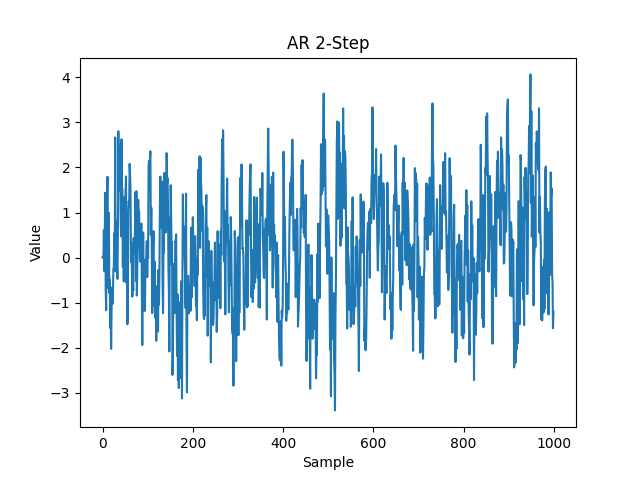
Auto Regressive:

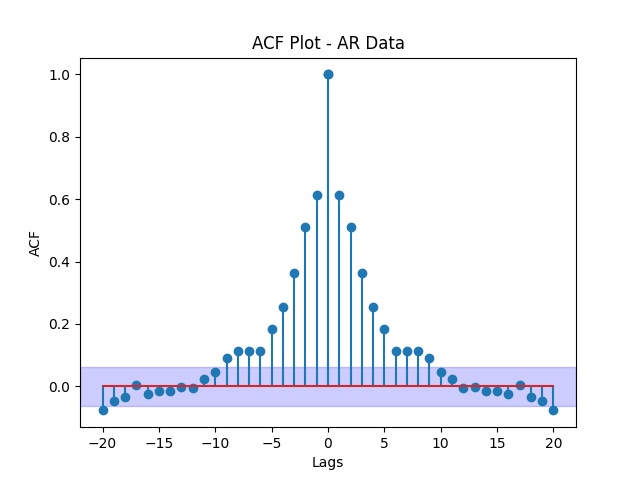


Moving Average:

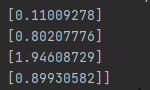


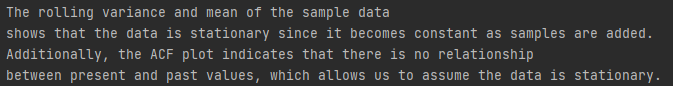
**Answers to Lab Questions:**

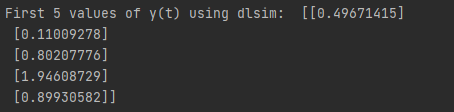
**1.**

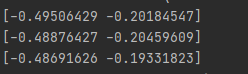
****

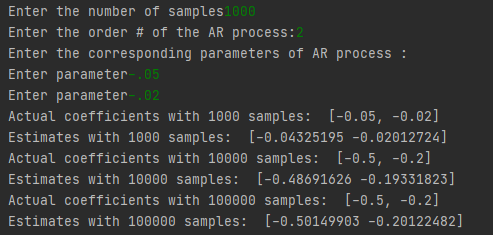


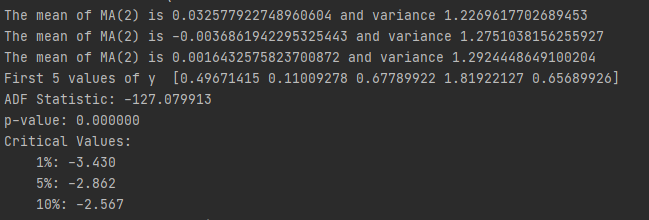




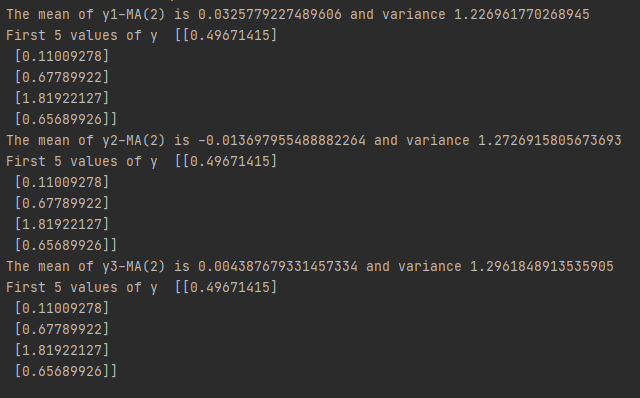
**2.** 

**3.** 

**4.** 

**5.** 

**The test statistic is less than the critical values, so this is not a stationary process.**

**6.** 

**Conclusion:**

The MA and AR processes can be used to predict future values based on previous values.

**Appendix:**

import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
import Toolbox  
from scipy import signal  
  
print("--------------------Question 1--------------------")  
# question 1  
sample\_count = 1000  
e = np.random.normal(0, 1, sample\_count)  
y = np.zeros(len(e))  
step = 2  
for t in range(sample\_count):  
 if t >= step:  
 y[t] = 0.5\*y[t-1] + 0.2\*y[t-2] + e[t]  
  
y = np.array(y)  
  
plt.plot(np.arange(sample\_count), y)  
plt.title('AR 2-Step')  
plt.ylabel('Value')  
plt.xlabel('Sample')  
plt.show()  
  
Toolbox.ACF(y, 20, 'AR Data')  
plt.show()  
  
Toolbox.cal\_rolling\_mean\_var(y, np.arange(sample\_count), metric='AR', unit='Sample')  
plt.show()  
  
print('First five values of y(t): ', y[:5])  
print('The rolling variance and mean of the sample data\nshows that the data is stationary since it becomes constant as samples are added. \nAdditionally, the ACF plot indicates that there is no relationship \nbetween present and past values, which allows us to assume the data is stationary.')  
  
print("--------------------Question 2--------------------")  
  
num = [1, 0, 0]  
den = [1, -0.5, -0.2]  
  
np.random.seed(42)  
T = 1000  
mean = 0  
std = np.sqrt(1)  
  
e = np.random.normal(mean, std, size=T)  
  
sys = (num, den, 1)  
\_, y2 = signal.dlsim(sys, e)  
  
print("First 5 values of y(t) using dlsim: ", y2[:5])  
  
print(f"The mean of y2-AR(2) is {np.mean(y2)} and variance {np.var(y2)}")  
  
  
  
# Question 3  
  
print("--------------------Question 3--------------------")  
  
np.random.seed(42)  
  
order = 2  
samples = 1000  
y = Toolbox.simulate\_AR(mean, std, samples)  
lse = Toolbox.least\_square\_estimate(y, samples, order)  
  
# 5000  
samples = 5000  
order = 2  
y = Toolbox.simulate\_AR(mean, std, samples)  
lse = Toolbox.least\_square\_estimate(y, samples, order)  
  
# 10000  
samples = 10000  
order = 2  
y = Toolbox.simulate\_AR(mean, std, samples)  
lse = Toolbox.least\_square\_estimate(y, samples, order)  
  
  
  
# Question 4  
  
print("--------------------Question 4--------------------")  
  
# 1000 samples  
samples = int(input("Enter the number of samples"))  
order = int(input("Enter the order # of the AR process:"))  
print("Enter the corresponding parameters of AR process :")  
ARparam = [float(input("Enter parameter")) for i in range(order)]  
  
LSE = Toolbox.generalized\_least\_square\_estimate(samples, order, ARparam)  
print("Actual coefficients with 1000 samples: ", ARparam)  
print("Estimates with 1000 samples: ", LSE)  
  
# 10000 samples  
LSE = Toolbox.generalized\_least\_square\_estimate(10000, 2, [-0.5, -0.2])  
print("Actual coefficients with 10000 samples: ", [-0.5, -0.2])  
print("Estimates with 10000 samples: ", LSE)  
  
# 100000 samples  
LSE = Toolbox.generalized\_least\_square\_estimate(100000, 2, [-0.5, -0.2])  
print("Actual coefficients with 100000 samples: ", [-0.5, -0.2])  
print("Estimates with 100000 samples: ", LSE)  
  
  
  
# Question 5 a  
  
print("--------------------Question 5--------------------")  
  
y = Toolbox.simulate\_MA(1000)  
  
# Question 5 b  
  
plt.figure(figsize=[10, 5])  
plt.plot(y)  
plt.xlabel('Sample #')  
plt.ylabel('y(t)')  
plt.title('Simulated MA(2) Series')  
plt.grid()  
plt.show()  
  
# Question 5 c  
  
print(f"The mean of MA(2) is {np.mean(y)} and variance {np.var(y)}")  
Toolbox.ACF(y, 20, 'Simulated MA(2) Series')  
plt.show()  
  
# Question 5 d  
  
# ACF of the series with 10000  
y = Toolbox.simulate\_MA(10000)  
print(f"The mean of MA(2) is {np.mean(y)} and variance {np.var(y)}")  
Toolbox.ACF(y, 20, 'Simulated MA(2) Series - 10000')  
plt.show()  
  
# ACF of the series with 100000  
y = Toolbox.simulate\_MA(100000)  
print(f"The mean of MA(2) is {np.mean(y)} and variance {np.var(y)}")  
Toolbox.ACF(y, 20, 'Simulated MA(2) Series - 100000')  
plt.show()  
  
# Question 5 e  
  
print("First 5 values of y ", y[:5])  
  
# Question 5 f  
  
Toolbox.ADF\_Cal(y)  
  
  
  
# Question 6  
  
print("--------------------Question 6--------------------")  
  
np.random.seed(42)  
num = [1, 0.5, 0.2]  
den = [1, 0, 0]  
  
e = np.random.normal(size=1000)  
sys = (num, den, 1)  
\_, y1 = signal.dlsim(sys, e)  
  
e = np.random.normal(size=10000)  
sys = (num, den, 1)  
\_, y2 = signal.dlsim(sys, e)  
  
e = np.random.normal(size=100000)  
sys = (num, den, 1)  
\_, y3 = signal.dlsim(sys, e)  
  
print(f"The mean of y1-MA(2) is {np.mean(y1)} and variance {np.var(y1)}")  
print("First 5 values of y ", y1[:5])  
Toolbox.ACF(y1, 20, 'Simulated MA(2) Series - 1000')  
plt.show()  
  
print(f"The mean of y2-MA(2) is {np.mean(y2)} and variance {np.var(y2)}")  
print("First 5 values of y ", y1[:5])  
Toolbox.ACF(y2, 20, 'Simulated MA(2) Series - 10000')  
plt.show()  
  
print(f"The mean of y3-MA(2) is {np.mean(y3)} and variance {np.var(y3)}")  
print("First 5 values of y ", y1[:5])  
Toolbox.ACF(y3, 20, 'Simulated MA(2) Series - 100000')  
plt.show()