DATS 6313 – Time Series Analysis & Modeling

Instructor: Reza Jafari

Lab #3

Bradley Reardon

2/8/2022

1 – Abstract:

This lab pertains to implementing the Autocorrelation function program using python and applying it to real world data by accessing time series datasets regarding stock value for 6 major companies using the yahoo API.

2 – Introduction:

This experiment was performed to increase understanding of what the autocorrelation metric represents, how to interpret it, and what it looks like visually when plotted amongst the data points.

3 – Method, Theory, and Procedures:

Autocorrelation measures the linear relationship between lagged values of time series. The notation used for autocorrelation is τk which shows the linear relationship between yt and yt−k . Ry (τ ) is an estimated autocorrelation for stationary time series y(t). It just depends on the lagged values of time series. The following formula is used to find Ry (τ ):

Text

Description automatically generated

In the first question, we manually find the autocorrelation of the provided y(t): [3, 9, 27, 81,243] and calculate the ACF for lag 0, 1, 2, 3, and 4 (𝑟𝑦(0), 𝑟𝑦(1), 𝑟𝑦(2), 𝑟𝑦(3), 𝑟𝑦(4) ). Next, we create a function in python to calculate the ACF and plot it accordingly. The following figures represent the functions used to find and plot ACF:

Text

Description automatically generated

The data used to experiment with calculating and plotting the ACF was gathered by accessing stock value information using the yahoo API. The company stocks and code to access the API are as follows:

stocks = ['AAPL','ORCL', 'TSLA', 'IBM','YELP', 'MSFT']

import pandas\_datareader as web web.DataReader('AAPL', data\_source='yahoo', start=, end=)

4 – Answers to Lab Questions:

1. A computer screen capture

   Description automatically generated with medium confidence

Timeline

Description automatically generated with low confidence

Please don’t dock points for manually calculating question 1 in pycharm 😊

1. The following scatter and histogram plots represent the white noise data generated:

Chart, scatter chart

Description automatically generated Chart, histogram

Description automatically generated

1. The following images show the code used to calculate and plot the ACF as well as the plots asked for:

Chart, line chart

Description automatically generatedChart

Description automatically generated

Text

Description automatically generated

The scatter and histogram plots show that the data is normalized. The ACF plot shows that there is no correlation between the error of each plotted point from the white noise data.

1. Graphical user interface, chart

   Description automatically generatedChart, bar chart

   Description automatically generated
2. 1. If the ACF is slowly decaying, that means future values of the series are correlated / heavily   
   affected by past values.
   1. Data is non-stationary.

2. If the ACF is fast decaying, that means future values of the series are not correlated / heavily   
affected by past values.

* 1. Data is stationary.

5 – Conclusion:

The ACF is beneficial in time series analysis as it allows us to measure the linear relationship between lagged values of time series. It is helpful in determining if data is stationary or not.

6 – Appendix

import numpy as np  
import matplotlib.pyplot as plt  
import pandas\_datareader as web  
from datetime import datetime  
  
# question 1  
y1 = 3  
y2 = 9  
y3 = 27  
y4 = 81  
y5 = 243  
ymean = 72.6  
  
r0 = (y1 - ymean)\*\*2 // (y1 - ymean)\*\*2  
r1 = ((y2-ymean)\*(y1-ymean) + (y3-ymean)\*(y2-ymean) + (y4-ymean)\*(y3-ymean) + (y5-ymean)\*(y4-ymean)) / ((y1-ymean)\*\*2+(y2-ymean)\*\*2+(y3-ymean)\*\*2+(y4-ymean)\*\*2+(y5-ymean)\*\*2)  
r2 = ((y3-ymean)\*(y1-ymean) + (y4-ymean)\*(y2-ymean) + (y5-ymean)\*(y3-ymean)) / ((y1-ymean)\*\*2 + (y2-ymean)\*\*2 + (y3-ymean)\*\*2 + (y4-ymean)\*\*2 + (y5-ymean)\*\*2)  
r3 = ((y4-ymean)\*(y1-ymean) + (y5-ymean)\*(y2-ymean)) / ((y1-ymean)\*\*2 + (y2-ymean)\*\*2 + (y3-ymean)\*\*2 + (y4-ymean)\*\*2 + (y5-ymean)\*\*2)  
r4 = ((y5-ymean)\*(y1-ymean)) / ((y1-ymean)\*\*2 + (y2-ymean)\*\*2 + (y3-ymean)\*\*2 + (y4-ymean)\*\*2 + (y5-ymean)\*\*2)  
print(r0, r1, r2, r3, r4)  
  
# question 2  
x = np.random.normal(loc=0, scale=1, size=1000)  
y = np.arange(0,1000)  
  
plt.scatter(x,y)  
plt.title('WN - Scatter')  
plt.show()  
  
plt.hist(x)  
plt.title('WN - Histogram')  
plt.show()  
  
# question 3  
def autocorrelation(x):  
 n = len(x)  
 variance = x.var()  
 x = x - x.mean()  
 r = np.correlate(x, x, mode='full')[-n:]  
 assert np.allclose(r, np.array([(x[:n - k] \* x[-(n - k):]).sum() for k in range(n)]))  
 result = r / (variance \* (np.arange(n, 0, -1)))  
 return result  
  
def autocorrelation\_plot(x, lag=1):  
 n = len(x)  
 variance = x.var()  
 x = x - x.mean()  
 r = np.correlate(x, x, mode='full')[-n:]  
 assert np.allclose(r, np.array([(x[:n - k] \* x[-(n - k):]).sum() for k in range(n)]))  
 result = r / (variance \* (np.arange(n, 0, -1)))  
  
 plt.acorr(result, maxlags = lag)  
 plt.title('Autocorrelation')  
 plt.ylabel('Magnitude')  
 plt.xlabel('Lags')  
 plt.grid(True)  
 plt.show()  
  
  
autocorrelation\_plot(np.array([3,9,27,81,243]), lag=4)  
autocorrelation\_plot(x, lag=20)  
  
'''  
The scatter and histogram plots show that the data is normalized  
The ACF plot shows that there is no correlation between the error   
of each plotted point from the white noise data.  
'''  
  
# question 4  
start = datetime(2000, 1, 1)  
end = datetime(2022, 2, 5)  
  
AAPL = web.DataReader('AAPL', data\_source='yahoo', start=start, end=end)  
ORCL = web.DataReader('ORCL', data\_source='yahoo', start=start, end=end)  
TSLA = web.DataReader('TSLA', data\_source='yahoo', start=start, end=end)  
IBM = web.DataReader('IBM', data\_source='yahoo', start=start, end=end)  
YELP = web.DataReader('YELP', data\_source='yahoo', start=start, end=end)  
MSFT = web.DataReader('MSFT', data\_source='yahoo', start=start, end=end)  
  
print(AAPL.head())  
#print(np.array(AAPL['Close']))  
#print(autocorrelation(np.array(AAPL['Close'])))  
  
  
fig, ax = plt.subplots(3, 2)  
ax1, ax2, ax3, ax4, ax5, ax6 = ax.flatten()  
fig.suptitle('Close Value of Stock: 2000-01-01 - Present')  
ax1.plot(np.arange(len(AAPL['Close'])), AAPL['Close'], label='AAPL')  
ax1.legend(loc='upper left')  
ax2.plot(np.arange(len(ORCL['Close'])), ORCL['Close'], label='ORCL')  
ax2.legend(loc='upper left')  
ax3.plot(np.arange(len(TSLA['Close'])), TSLA['Close'], label='TSLA')  
ax3.legend(loc='upper left')  
ax4.plot(np.arange(len(IBM['Close'])), IBM['Close'], label='IBM')  
ax4.legend(loc='upper left')  
ax5.plot(np.arange(len(YELP['Close'])), YELP['Close'], label='YELP')  
ax5.legend(loc='upper left')  
ax6.plot(np.arange(len(MSFT['Close'])), MSFT['Close'], label='MSFT')  
ax6.legend(loc='upper left')  
fig.text(0.5, 0.04, 'Days', ha='center')  
fig.text(0.04, 0.5, 'Stock Value', va='center', rotation='vertical')  
  
plt.show()  
  
fig1, ax = plt.subplots(3, 2)  
ax1, ax2, ax3, ax4, ax5, ax6 = ax.flatten()  
fig1.suptitle('ACF of Close Value of Stock: 2000-01-01 - Present')  
ax1.acorr(autocorrelation(np.array(AAPL['Close'])), maxlags = 40, label='AAPL')  
ax1.legend(loc='upper left')  
ax2.acorr(autocorrelation(np.array(ORCL['Close'])), maxlags = 40, label='ORCL')  
ax2.legend(loc='upper left')  
ax3.acorr(autocorrelation(np.array(TSLA['Close'])), maxlags = 40, label='TSLA')  
ax3.legend(loc='upper left')  
ax4.acorr(autocorrelation(np.array(IBM['Close'])), maxlags = 40, label='IBM')  
ax4.legend(loc='upper left')  
ax5.acorr(autocorrelation(np.array(YELP['Close'])), maxlags = 40, label='YELP')  
ax5.legend(loc='upper left')  
ax6.acorr(autocorrelation(np.array(MSFT['Close'])), maxlags = 40, label='MSFT')  
ax6.legend(loc='upper left')  
fig1.text(0.5, 0.04, 'ACF', ha='center')  
fig1.text(0.04, 0.5, 'Magnitude', va='center', rotation='vertical')  
  
plt.show()