The George Washington University

Laboratory Three:

Implementing Autocorrelation Function in Time Series Anlaysis

Fernando Zambrano

DATS 6450: Multivariate Modeling

Dr. Reza Jafari

12 February 2020

Abstract:

Estimate the autocorrelation function (ACF) of simple series by hand to better understand the formula and the effect of different lags. Create a custom function, auto_cor, in Python to calculate the ACF of any series and given lags. Implement auto_cor on generated white noise data to demonstrate the lack of autocorrelation on randomly initiated data. Additionally, auto_cor is implemented on a small company's financial time series data to illustrate how ACF can show characteristics in the data that may be missed visually or by other statistical measures such as an Augmented Dickey-Fuller (ADF) test for establishing stationarity.

Introduction:

Often, one of the first steps in any data analysis is performing regression analysis. However, one of the assumptions of regression analysis is that the data has no autocorrelation. This can be frustrating because applying regression analysis on data with autocorrelation will lead to misleading results. Additionally, some time series forecasting methods rely on the assumption that there isn't any autocorrelation in the residuals, the error between prediction and observation. Forecasting analysts often use the residuals to assess whether their model is a good fit but may ignore or forget to assess the autocorrelation in residuals. This mistake can mislead into believing that certain models are a good fit when in fact they are not. Inspecting for autocorrelation analysis can help uncover hidden patterns in data and help select the correct forecasting methods. Specifically, it helps identify seasonality and trend in time series data. Additionally, analyzing the autocorrelation function (ACF) is necessary for selecting the appropriate model for time series prediction.

Methods & Theory:

Correlation is a dimensionless measurement that quantifies the linear relationship between two independent features.

$$r = \frac{\sum(x_t - \bar{x})(y_t - \bar{y})}{\sqrt{\sum(x_t - \bar{x})^2}\sqrt{\sum(y_t - \bar{y})^2}}.$$

$$\hat{\tau}_k = \frac{\sum_{t=k+1}^{T}(y_t - \bar{y})(y_{t-k} - \bar{y})}{\sum_{t=1}^{T}(y_t - \bar{y})^2}$$

The numerator of this equation is the covariance between two variables. Covariance determines how linearly related two features are associated. The issue with why covariance is not used as the principle statistic to demonstrate the strength of a linear relationship is that it maintains the units of its features. This makes it difficult to draw comparisons between two features that have large differences in their spread. To compare two features without the drawback of dimensionality, the features need to be normalized. The denominator is the product of the square root of the squared variance of each feature. This process cancels out the dimensions of each feature and forces the

resulting value of *r* to fall between -1 and +1. The absolute value of r demonstrates the strength of the correlation between two variables. If the correlation is positive, then both features increase and decrease correspondingly. If the correlation is negative, then both features associate inversely. As one feature increases, then the other feature decreases and vice averse.

Autocorrelation, *T*, is the correlation of a series with itself with some amount of lag *k*, instead of another feature. Lag is the number of time steps between two points in a series. Autocorrelation ranges in values from -1 to + 1. The formula for calculating the autocorrelation with k lags is the autocorrelation function (ACF) which is below.

An example of calculating autocorrelation is to create a simple series, x(t) = [2,3,4] compared to itself with lag k = 0. It might be helpful to think of comparing two different series, x(t), and the lagged version of x(t), y(t). Since k = 0, there is no lag and the autocorrelation would be estimating the correlation of x(t) = [2,3,4] with y(t) = [2,3,4]. The resulting *T* is 1.0 since there is a perfect linear relationship between two series with the same value at each respective time step. If k =1, then there would be a lag of 1 timestep and *T* would be estimating the correlation between x(t) = [2,3] and y(t) = [3,4]. The resulting *T* is no longer 1.0 since the exact points are no longer being compared. Instead, the first value in x(t), 2, is compared to the value that immediately succeeds it, which is the first value in y(t), 3. This demonstrates how to implement lag by one step. If k = 2, then the autocorrelation between x(t) and y(t) is calculated by estimating *T* between x(t) = [2] and y(t) = [4] because 4 is two time-steps behind 2. A correlogram shows the correlation of a series at multiple lag points and is also known as an ACF plot. The visual interpretation of the series' autocorrelation can provide insight into underlying ways in which the data behaves over time that may be missed in simply plotting the raw data.

Visualizing an ACF plot is not only limited to the dataset itself, but also the residuals of a regression model. If the residuals resemble the steps of a person going on a random walk, then the residuals are considered to be "white noise." When the residuals are white noise, then all of the underlying patterns in the time series have been captured by the model. The residuals of white noise should have an autocorrelation of 1 at lag = 0, and autocorrelation near 0 at all other lag points. However, when models do not fully capture the underlying patterns in the data, the residuals will show obvious signs of high autocorrelation. These kinds of residuals are considered as "colored" or "pink" noise. Hence, ACF plots illustrate a method to run diagnostics on models to make sure there is no unincorporated information in the model.

Two methods will be explored to distinguish whether a dataset is stationary or not. The first option is to visualize the data by plotting its features over time and looking for obvious patterns. This method is highly subjective on the reader's interpretation of the data which may result in an inaccurate conclusion of stationarity. However, it offers the analyst into what kind of patterns the data may follow.

Three patterns that can help determine the stationarity of the dataset: trend, seasonality, and cyclicity. The trend is the "direction" which the data is moving towards. Is it mainly increasing or decreasing over time? If there is an obvious trend, then it is likely the data is non-stationary as the mean is increasing at a varying rate over time. Seasonality is the effect of the specific points

in time that change the behavior of the data. These changes happen in fixed and known frequencies which allows for predictable changes. Cyclicity refers to the data exhibiting cycle of rises and falls that occur in irregular frequencies. Cyclic behavior can be clear in hindsight but becomes more unpredictable in the long-term future. The differences between seasonal and cyclic is that seasonal changes occur at constant length, whereas changes in cyclic vary. Additionally, for reemphasis, seasonality is dependent directly with time, while cyclicity is a result of previous values. Furthermore, seasonality and cyclicity are not clear signs of the data being non-stationary, as long as the change in the variance remains constant over time the data will remain stationary

A popular option to determine if a dataset is stationary is to perform a statistical hypothesis test which determines the likelihood of the dataset being non-stationary. The Augmented Dickey-Fuller (ADF) test, or "unit root test" makes a strong assumption that dataset in hand is non-stationary. This test sets up two hypotheses:

Null Hypothesis (H0): if failed to be rejected, then the time-series has a unit root and the data is non-stationary.

The Alternative Hypothesis (H1): if the null hypothesis is rejected, then there is no unit root and the data is stationary.

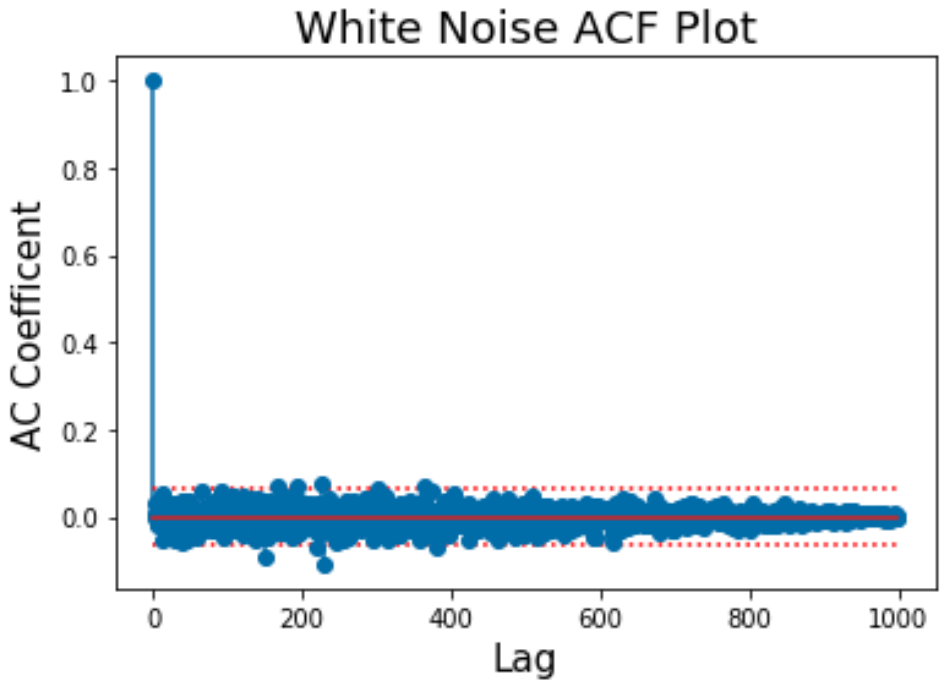
In order to reject the null hypothesis, there must be strong evidence to suggest that the likelihood of the data being non-stationary is significantly unlikely. This is done by setting high confidence levels which are determined by the predetermined critical values. The standard practice is to have between a 95% - 99% confidence level. A 0.05 and 0.01 critical values correspond with a 95% and 99% confidence level, respectively. The ADF test will result with a p-value which is compared to the set critical value. If the p-value is less than the critical value, then there is significantly strong evidence to suggest that the null hypothesis (non-stationary) be rejected in favor of the alternative (stationary). Conversely, if the p-value is greater than the critical value, then there is not enough significantly strong evidence to support rejecting the null hypothesis

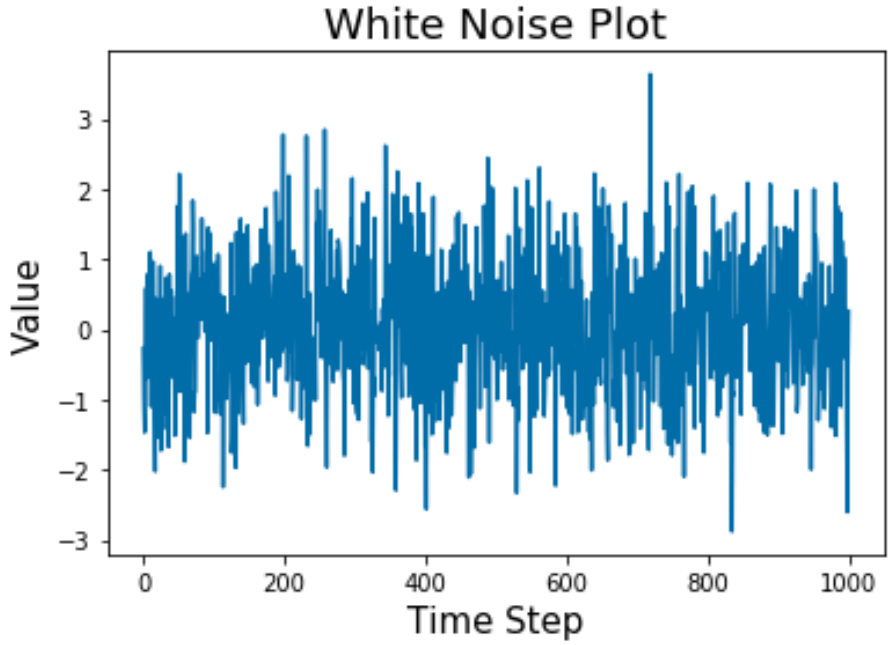
Implementation and Results:

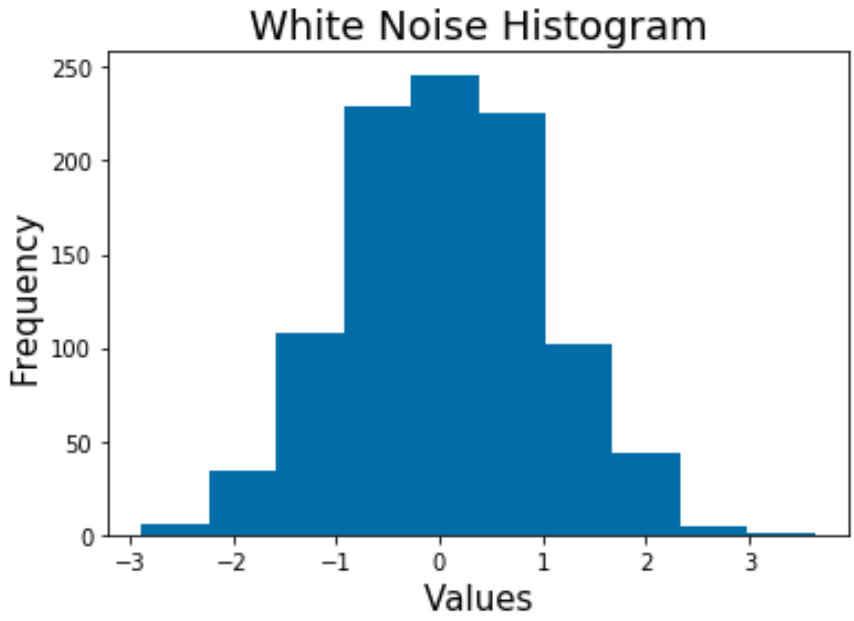
The formula below estimates the autocorrelation function (ACF) of a time series. This formula is used to calculate the autocorrelation at different lags for the following series: y(t) = [3,9,81,27,243] by hand. The results of these calculations are located in the appendix. Section before the python code section. The ACF formula was also implemented in Python to create a customized function, **auto_cor**, for estimating the ACF of series without the use of built statistical functions.

$$\hat{\tau}_k = \frac{\sum_{t=k+1}^{T}(y_t - \overline{y})(y_{t-k} - \overline{y})}{\sum_{t=1}^{T}(y_t - \overline{y})^2}$$

The function auto_cor was tested on randomly generated white noise data, which has a mean of 0 and a standard deviation of 1. There are 1,000 samples in this series. The resulting ACF plot is below, followed by white noise vs time and its histogram.

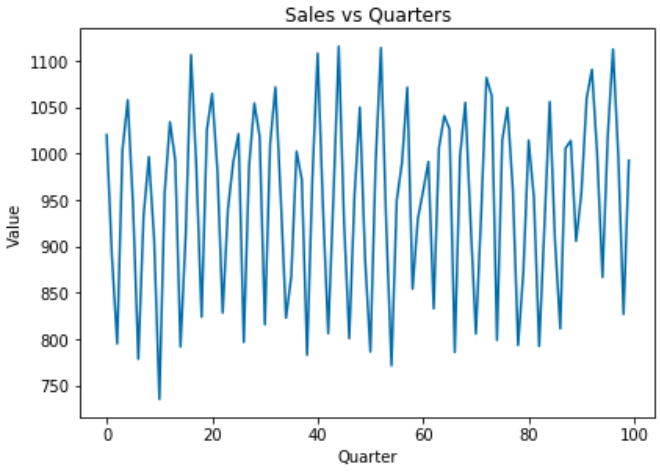


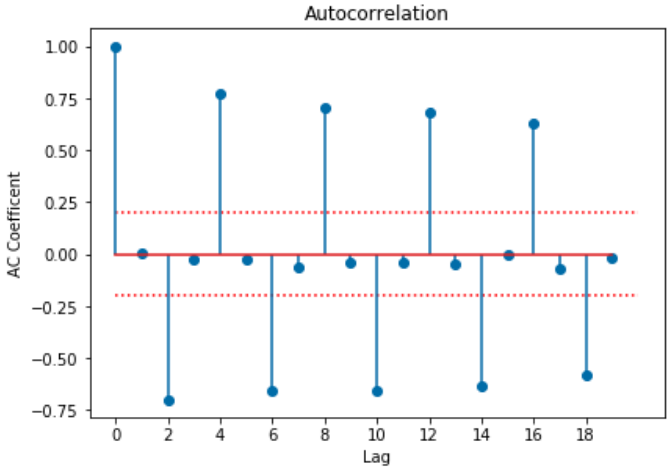




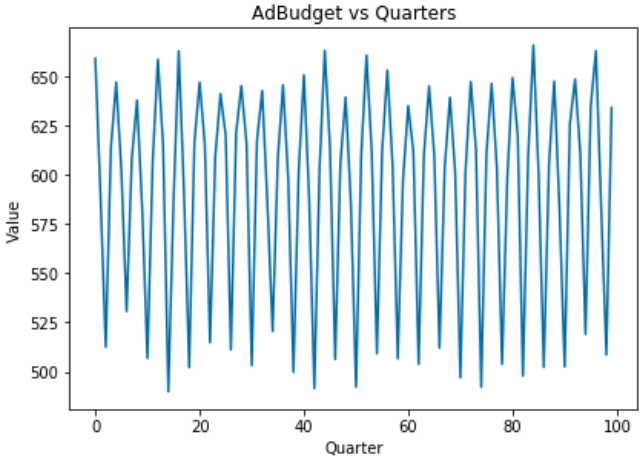
ACF demonstrates the generated white noise data is not autocorrelated since the entire dataset seems to be centered around 0 autocorrelation. The histogram demonstrates that the data is normally distributed which means the data is likely to be stationary. The time plot demonstrates that there is no obvious signs of trend, seasonality, nor cyclic behavior. Which also provides evidence that the data is possibly stationary.

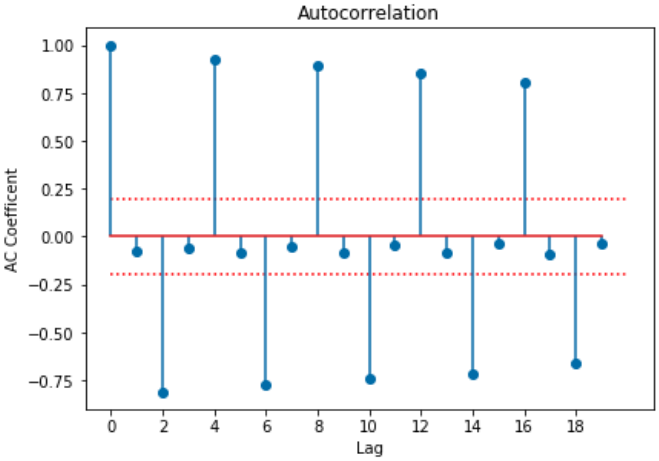
Moving beyond white noise, the auto_cor function was applied on the financial data of a small business. The ACF of sales, advertisement budget, and GDP features were calculated with a lag = 20 and plotted next to a graph of each feature over quarters. An ADF test was also applied on each feature to determine to what extent it may be considered stationary or non-stationary. The results are below.



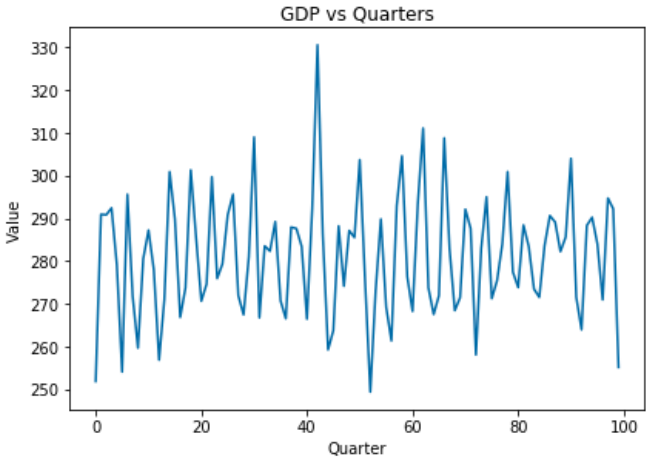


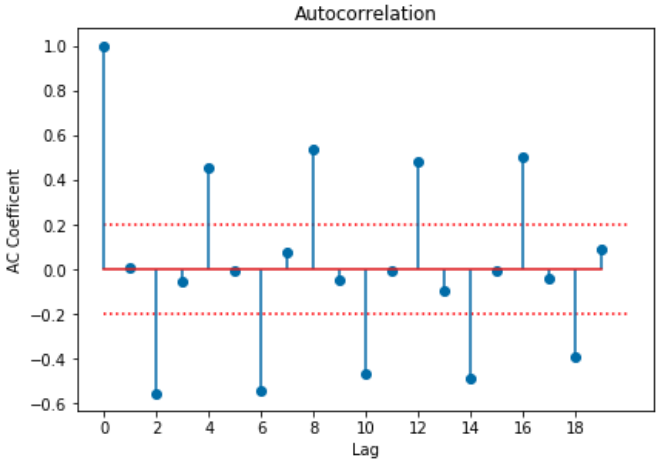
```
The mean of this dataset is:  948.737
The total variance for this dataset is:  955695.7331000001
ADF Statistic: -3.262755
p-value: 0.016628
Used lags: 9.000000
Critical Values:
        1%: -3.505190
        5%: -2.894232
        10%: -2.584210
```






```
The mean of this dataset is:  591.933
The total variance for this dataset is:  292357.20109999995
ADF Statistic: -2.758605
p-value: 0.064434
Used lags: 7.000000
Critical Values:
        1%: -3.503515
        5%: -2.893508
        10%: -2.583824
```





```
The mean of this dataset is:  281.18300000000005
The total variance for this dataset is:  20444.581100000003
ADF Statistic: -3.227577
p-value: 0.018443
Used lags: 7.000000
Critical Values:
        1%: -3.503515
        5%: -2.893508
        10%: -2.583824
```

Sales and GDP data are considered stationary series by the ADF test since they both resulted with p-values below 0.05. On the other hand, AdBudget failed to reject the null hypotheses of the ADF test because its p-value was above 0.05 which makes it non-stationary. There is a clear oscillation pattern of high positive autocorrelation every multiple of 4 lags and high negative autocorrelation every 4 lags starting at lag 2. This makes senses since the data is quarterly so that every 4 lags are one year apart. One possible correlation between the stationarity and degree of autocorrelation of time series is how quickly the magnitude of autocorrelation decays. The ACF plots of Sales and GDP illustrate their autocorrelation lower significantly faster than that of ACF plot of AdBudget. Otherwise, there is no clear correlation or relationship between the autocorrelation and that stationarity of a time series.

Conclusion:

Overall, the implemented auto_cor python function allows for quick estimations of autocorrelation of a time-series at any given lag within the range of the dataset. This method offers quicker calculations than by hand since the formula requires a summation across the entire series which can take a long time for just couple of lags if the dataset is large. Additionally, creating an ACF plot of randomly generated normally distributed data illustrates the fundamental characteristics of white noise. Residuals of a regression model should ideally be normally distributed and with an ACF hovering around 0 just like white noise. Furthermore, through the auto-corr function, the financial data of the small business is shown to be moderately to highly autocorrelated. However, there is no clear association between the degree of autocorrelation and whether or not the time-series is stationary or non-stationary.

Appendix:

$$\tau_k = \frac{\sum_{t=k+1}^{T} (y_t - \bar{y})(y_{t-k} - \bar{y})}{\sum_{t=1}^{T} (y_t - \bar{y})^2}$$

$y(t) = [3, 9, 27, 81, 243]$

$$\bar{y} = \frac{3+9+27+81+243}{5} = 72.6$$

$$\boxed{\bar{y} = 72.6}$$

y_t	$y_t - \bar{y}$		$(y_t - \bar{y})^2$	
3	$3 - 72.6 =$	-69.6	$(-69.6)^2 =$	4844.16
9	$9 - 72.6 =$	-63.6	$(-63.6)^2 =$	4044.96
27	$27 - 72.6 =$	-45.6	$(-45.6)^2 =$	2079.36
81	$81 - 72.6 =$	8.4	$(8.4)^2 =$	70.56
243	$243 - 72.6 =$	170.4	$(170.4)^2 =$	29036.16
			Sum =	40075.2

$$\tau_0 = \text{Lag} = 0 = \frac{\sum[(-69.6)^2 + (-63.6)^2 + (-45.6)^2 + (8.4)^2 + (170.4)^2]}{40075.2}$$

$$\tau_0 \quad \cancel{\tau_0 = *} = \frac{40075.2}{40075.2} = 1 \qquad \boxed{\tau_0 = 1.000}$$

$$\tau_1 = \text{Lag} = 1 = \frac{\sum[(-69.6 * 0) + (-63.6 * -69.6) + (-45.6 * -63.6) + (8.4 * -45.6) + (170.4 * 8.4)]}{40075.2}$$

$$= \frac{0 + 4426.56 + 2900.16 + (-383.04) + 1431.36}{40075.2}$$

$$\tau_1 = 8375.04 / 40075.2 \rightarrow \boxed{0.2089\,83 = \tau_1}$$

$$T_2 = Lag=2 = \frac{\Sigma[(-69.6*0)+(-63.6*0)+(-45.6*-69.6)+(8.4*-63.6)+(170.4*-45.6)}{40075.2}$$

$$T_2 = \frac{0+0+3173.76+(-534.24)+(-7770.24)}{40075.2}$$

$$T_2 = \frac{-5130.72}{40075.2} = \boxed{-0.128027 = T_2}$$

$$T_3 = lag=3 = \frac{\Sigma[(-69.6*0)+(-63.6*0)+(-45.6*0)+(8.4*-69.6)+(170.4*-63.6)}{40075.2}$$

$$T_3 = \frac{0+0+0+(-584.64)+(-10857.44)}{40075.2}$$

$$T_3 = \frac{-11422.08}{40075.2} = \boxed{-0.285016 = T_3}$$

$$T_4 = lag=4 = \frac{\Sigma[(-69.6*0)+(-63.6*0)+(-45.6*0)+(8.4*0)+(170.4*-69.6)}{}$$

$$T_4 = \frac{0+0+0+0+(-11859.84)}{40075.2} \rightarrow \frac{-11859.84}{40075.2}$$

$$\boxed{T_4 = -0.295940}$$

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.stattools import adfuller
import os

def ADF_Cal(x):
    result = adfuller(x)
    print("ADF Statistic: %f" %result[0])
    print("p-value: %f" %result[1])
    print("Used lags: %f" %result[2])
    print("Critical Values:")
    for key, value in result[4].items():
        print('\t%s: %3f' % (key,value))

# 1
# Equation for autocorrelation
# k = number of lags. t = number of samples

# 2
y = [3,9,27,81,243]
# Calculate mannually the t0,t1 etc t5. k = 5
# In appendix

# 3
# Create a white noise graph with mean = 0 and standard devation = 1 with N = 1000
# sd * (np.random.randn(# of samples) + mean)


s = 1
u = 0
N = 1000

X = s * np.random.randn(N) + u


# 4
```

```python
# Write a python code to calculate the autocorrelation function

def auto_corr(data, lags):
    # convert input data into a numpy array
    data = np.array(data)
    # acf will store the autocorreltion coefficent at each lag interval
    # the first datapoint is always 1.0 since anything correlated with itsself is = 1
    acf = [1.0]
    # calculate the mean for the entire dataset
    y_bar = data.mean()
    print("The mean of this dataset is: ", y_bar)
    # subtract the mean from each observation
    yy_bar = data - y_bar
    # clacualte the total variance for the data set
    total_variance = sum(np.square(yy_bar))
    print("The total variance for this dataset is: ", total_variance)
    # perform a forloop over the dataset with the desired number of lags
    # range is 1,lags b/c the first iteration calcualtes T1
    for i in range(1, lags):
        # first nparray is removing the last element each iteration
        yy_bar_bottom = yy_bar[:-i]
        # second nparray removes the first element each interation
        yy_bar_top = yy_bar[i:]
        # take the sum of of the product of each nparray each iteration
        yy = sum(yy_bar_top * yy_bar_bottom)
        # divide the sum by total variance and append to resulting acf list
        acf.append(yy / total_variance)
    return acf


# 4a
# Plot the ACF  for the white noise data using the stem command
wn_acf = auto_corr(X,len(X))

# Graph white noise ACF
figure = plt.stem(wn_acf,use_line_collection=True)
#plt.figure(figsize=(20,20))
```

```
plt.xlabel('Lag', fontsize=15)
# Incorporate 95% ACF white noise threshold
# +-2/ squareroot of T(1000) = 0.0632
plt.hlines(0.063245,0,1000,"r",linestyles="dotted")
plt.hlines(-0.063245,0,1000,"r",linestyles="dotted")
plt.ylabel('AC Coefficent', fontsize=15)
plt.title('White Noise ACF Plot',fontsize=18)
plt.show()

# 4b
# Graph white noise vs time-step and histogram
figure = plt.plot(X)
#plt.figure(figsize=(20,10))
plt.xlabel('Time Step', fontsize=15)
plt.ylabel('Value', fontsize=15)
plt.title('White Noise Plot',fontsize=18)
plt.show()

# 4b
# Graph white noise vs time-step and histogram
figure = plt.hist(X)
#plt.figure(figsize=(20,10))
plt.xlabel('Values', fontsize=15)
plt.ylabel('Frequency', fontsize=15)
plt.title('White Noise Histogdram',fontsize=18)
plt.show()

#4c
# Write down observations
# ACF demonstrates the data is not autocorrelated since the entire dataset seems to be centered around 0
Autocorrelation
# The Histogram demonstrates that the data is normally distributed which means the data is likely to be stationary
# The Time plot demonstrates that there is no obvious signs of trend, seasonality, nor cyclic behavior
# This also provided evidence that the data is possibley stationary.

# 5
# load the tute1.csv dataset
```

```
os.listdir()
df = pd.read_csv("tute1.csv")
#plt.title("Scatter Plot of " + str(x) + " and " + str(y) + " with r = {}".format(r))

sales_acf = auto_corr(df.Sales,20)
adf_sales = ADF_Cal(df.Sales)

# 5a
# Plot ACF for Sales, and Sales vs Time next to each other

plt.figure(figsize=(12, 8))
plt.subplot(2, 2, 1)
plt.title('Sales vs Quarters')
plt.ylabel("Value")
plt.xlabel("Quarter")
plt.plot(df.Sales)

plt.subplot(2, 2, 2)
plt.title('Auto-correlation')
plt.xlabel('Lag')
plt.ylabel('AC Coefficent')
# Incorporate 95% ACF white noise threshold
# +-2/ squareroot of T(100) = 0.2
plt.hlines(0.2,0,20,"r",linestyles="dotted")
plt.hlines(-0.2,0,20,"r",linestyles="dotted")
plt.xticks(np.arange(0, 20, step=2))
plt.stem(sales_acf,use_line_collection=True)

#plt.text(-7,1.2,"ADF Test Results = {}".format(x))
plt.tight_layout()
plt.show()

# 5b
# Plot acf for AdBudget and AdBudget vs Time

adb_acf = auto_corr(df.AdBudget,20)
adf_adb = ADF_Cal(df.AdBudget)
```

```
plt.figure(figsize=(12, 8))
plt.subplot(2, 2, 1)
plt.title('AdBudget vs Quarters')
plt.ylabel("Value")
plt.xlabel("Quarter")
plt.plot(df.AdBudget)

plt.subplot(2, 2, 2)
plt.title('Auto-correlation')
plt.xlabel('Lag')
plt.ylabel('AC Coefficent')
# Incorporate 95% ACF white noise threshold
# +-2/ squareroot of T(100) = 0.2
plt.hlines(0.2, 0, 20, "r", linestyles="dotted")
plt.hlines(-0.2, 0, 20, "r", linestyles="dotted")
plt.xticks(np.arange(0, 20, step=2))
plt.stem(adb_acf, use_line_collection=True)

plt.tight_layout()
plt.show()

# 5c
## Plot ACF for GDP , and GDP vs Time next to each other

gdp_acf = auto_corr(df.GDP,20)
adf_gdp = ADF_Cal(df.GDP)

plt.figure(figsize=(12, 8))
plt.subplot(2, 2, 1)
plt.title('GDP vs Quarters')
plt.ylabel("Value")
plt.xlabel("Quarter")
plt.plot(df.GDP)

plt.subplot(2, 2, 2)
plt.title('Auto-correlation')
```

```
plt.xlabel('Lag')
plt.ylabel('AC Coefficent')
plt.xticks(np.arange(0, 20, step=2))
# Incorporate 95% ACF white noise threshold
# +-2/ squareroot of T(100) = 0.2
plt.hlines(0.2, 0, 20, "r", linestyles="dotted")
plt.hlines(-0.2, 0, 20, "r", linestyles="dotted")
plt.stem(gdp_acf, use_line_collection=True)

plt.tight_layout()
plt.show()
```

References:

Principles of Forecasting