THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

TIME SERIES MODELING & ANALYSIS

Instructor Name: Reza Jafari

Lab#: 3

Submitted by Dinesh Kumar Padmanabhan

Date: 30-sept-2020

ABSTRACT

In LAB #3, we learned concepts of correlation and auto correlation and their relationship w.r.t stationary and non-stationary data sets. ACF was calculated manually and plotted two sided. Using simple datasets, we plotted white noise, histogram and time series plots and made a comparison with auto correlation and time series plot.

INTRODUCTION

Correlation measures the extend of a linear relationship between two variables. Auto-correlation measures the linear relationship between lagged values of time series. The notation used for autocorrelation is τk which shoes the linear relationship between yt and yt−k . τk for stationary processes is time invariant. It just depends on the lagged values of time series.

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

where y(t) is the observation at time t and y is the sample mean of all observations

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*

Theory:

To Plot the Auto correlation plots for the given data set and determine how variables in the dataset are correlated.

Procedure:

I shall be looking at the variables through ACF and time series plots and infer about it in my analysis. And through my exploration I shall try to identify the how the variables are correlated and draw inferences.

The Dataset will be explored in following stages:

1. **Data Exploration (EDA)** – looking at continuous variables and making inferences about the data.
2. **Data Visualization** – Plotting scatter plots for the variables.
3. **Testing** – Running ACF to identify the correlation between them.

ANSWERS TO ASKED QUESTIONS

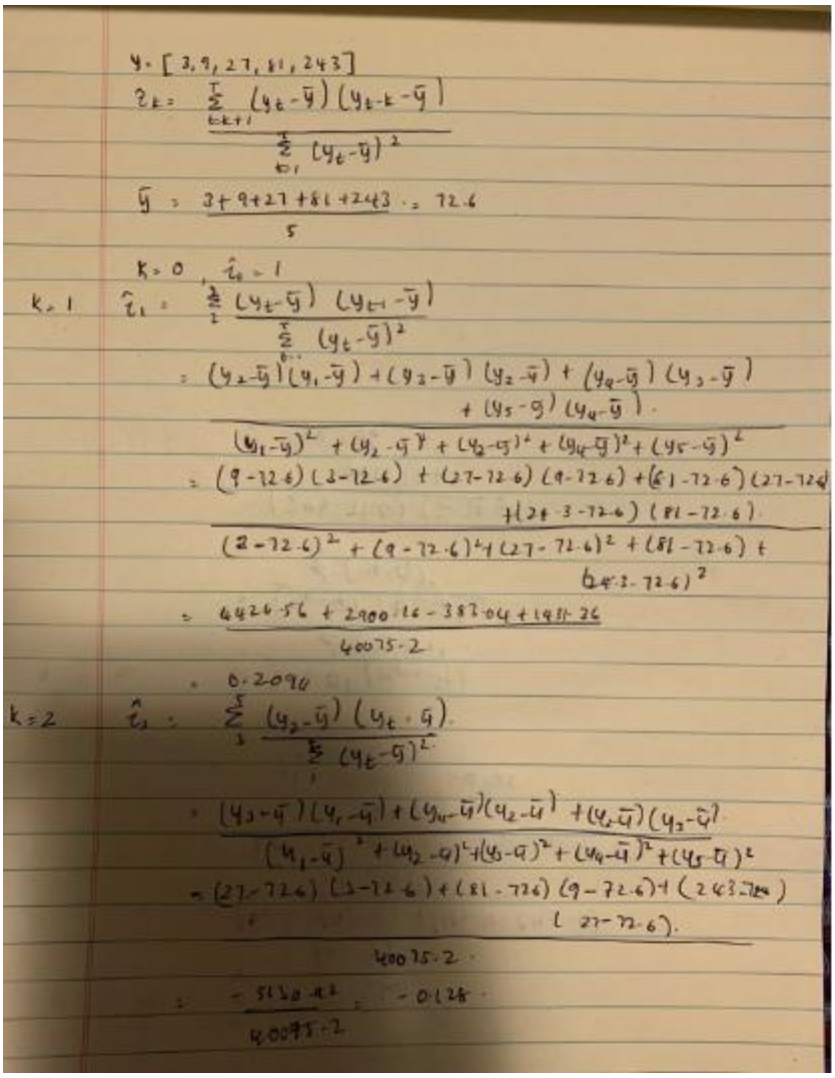
File - unknown

```
C:\ProgramData\Anaconda3\python.exe "C:\Program Files\JetBrains\PyCharm 2019.3.1\plugins\python\helpers\pydev\pydevconsole.py" --mode=client --port=54318

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/LAB3.py', wdir='C:/Users/nsree_000/Desktop/Python-Quiz/TIME SERIES')
The Mean of a white noise: 0.051539641633361985
The Standard Deviation of White Noise: 1.031411695249564
C:/Users/nsree_000/Desktop/Python-Quiz/TIME SERIES/LAB3.py:82: UserWarning: In Matplotlib 3.3 individual lines on a stem plot will be added as a LineCollection instead of individual lines. This significantly improves the performance of a stem plot. To remove this warning and switch to the new behaviour, set the "use_line_collection" keyword argument to True.
  plt.stem(range(-(k - 1), k), acfplotvals)
C:/Users/nsree_000/Desktop/Python-Quiz/TIME SERIES/LAB3.py:110: UserWarning: In Matplotlib 3.3 individual lines on a stem plot will be added as a LineCollection instead of individual lines. This significantly improves the performance of a stem plot. To remove this warning and switch to the new behaviour, set the "use_line_collection" keyword argument to True.
  ax1.stem(range(-(k-1),k), Salesacfplotvals)
C:\ProgramData\Anaconda3\lib\site-packages\pandas\plotting\_matplotlib\converter.py:103: FutureWarning: Using an implicitly registered datetime converter for a matplotlib plotting method. The converter was registered by pandas on import. Future versions of pandas will require you to explicitly register matplotlib converters.
```

Page 1 of 2

```

To register the converters:
    >>> from pandas.plotting import register_matplotlib_converters
    >>> register_matplotlib_converters()
  warnings.warn(msg, FutureWarning)
C:/Users/nsree_000/Desktop/Python-Quiz/TIME SERIES/LAB3.py:125: UserWarning: In Matplotlib 3.3 individual lines on a stem plot will be added as a LineCollection instead of individual lines. This significantly improves the performance of a stem plot. To remove this warning and switch to the new behaviour, set the "use_line_collection" keyword argument to True.
  ax1.stem(range(-(k-1),k), AdBudgetacfplotvals)
C:/Users/nsree_000/Desktop/Python-Quiz/TIME SERIES/LAB3.py:140: UserWarning: In Matplotlib 3.3 individual lines on a stem plot will be added as a LineCollection instead of individual lines. This significantly improves the performance of a stem plot. To remove this warning and switch to the new behaviour, set the "use_line_collection" keyword argument to True.
  ax1.stem(range(-(k-1),k), GDPacfplotvals)

```

```
#Using the Python program and using only the “numpy” and “matplotlib” library
perform the following tasks:
#%%==============================================================================
=# 1: Let suppose y vectors is given as y(t) = [3, 9, 27, 81,243].
# Without use of python or any other computer program, manually calculate the τ0,
τ1, τ2, τ3, τ4.
# Display the ACF (two sided) on a graph (no python).
# %%------------------------------------------------------------------------------
```



$y = [3, 9, 27, 81, 243]$

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

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

$k = 0, \quad \hat{\tau}_0 = 1$

$k = 1$

$$\hat{\tau}_1 = \frac{\sum_{2}^{5} (y_t - \bar{y})(y_{t-1} - \bar{y})}{\sum_{t=1}^{5} (y_t - \bar{y})^2}$$

$$= \frac{(y_2-\bar{y})(y_1-\bar{y}) + (y_3-\bar{y})(y_2-\bar{y}) + (y_4-\bar{y})(y_3-\bar{y}) + (y_5-\bar{y})(y_4-\bar{y})}{(y_1-\bar{y})^2 + (y_2-\bar{y})^2 + (y_3-\bar{y})^2 + (y_4-\bar{y})^2 + (y_5-\bar{y})^2}$$

$$= \frac{(9-72.6)(3-72.6) + (27-72.6)(9-72.6) + (81-72.6)(27-72.6) + (243-72.6)(81-72.6)}{(3-72.6)^2 + (9-72.6)^2 + (27-72.6)^2 + (81-72.6) + (243-72.6)^2}$$

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

$= 0.2090$

$k = 2$

$$\hat{\tau}_2 = \frac{\sum_{3}^{5} (y_t - \bar{y})(y_{t-2} - \bar{y})}{\sum_{1}^{5} (y_t - \bar{y})^2}$$

$$= \frac{(y_3-\bar{y})(y_1-\bar{y}) + (y_4-\bar{y})(y_2-\bar{y}) + (y_5-\bar{y})(y_3-\bar{y})}{(y_1-\bar{y})^2 + (y_2-\bar{y})^2 + (y_3-\bar{y})^2 + (y_4-\bar{y})^2 + (y_5-\bar{y})^2}$$

$$= \frac{(27-72.6)(3-72.6) + (81-72.6)(9-72.6) + (243-72.6)(27-72.6)}{40075.2}$$

$$= \frac{-5130.42}{40075.2} = -0.128$$

$$k=3 \quad \hat{r}_3 = \frac{\sum_{4}^{5} (y_t-\bar{y})(y_{t-3}-\bar{y})}{\sum_{1}^{5} (y_t-\bar{y})^2}$$

$$= \frac{(y_4-\bar{y})(y_1-\bar{y}) + (y_5-\bar{y})(y_2-\bar{y})}{(y_t-\bar{y})^2}$$

$$= \frac{(81-72.6)(3-72.6)(243-72.6)(9-72.6)}{40075.2}$$

$$= \frac{-11,422.08}{40075.2} = -0.285$$

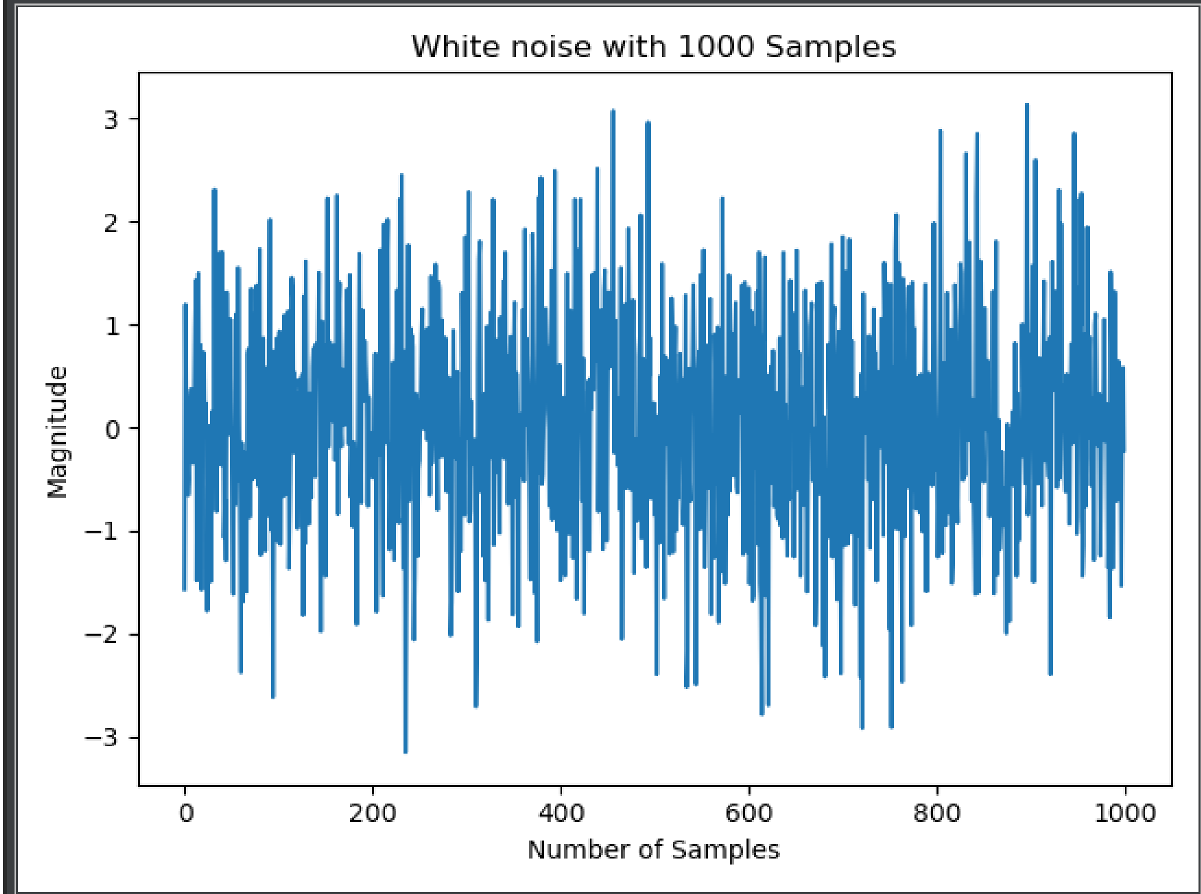
$$k=4 \quad \hat{r}_4 = \frac{\sum_{5}^{5} (y_t-\bar{y})(y_{t-4}-\bar{y})}{\sum_{1}^{5} (y_t-\bar{y})^2}$$

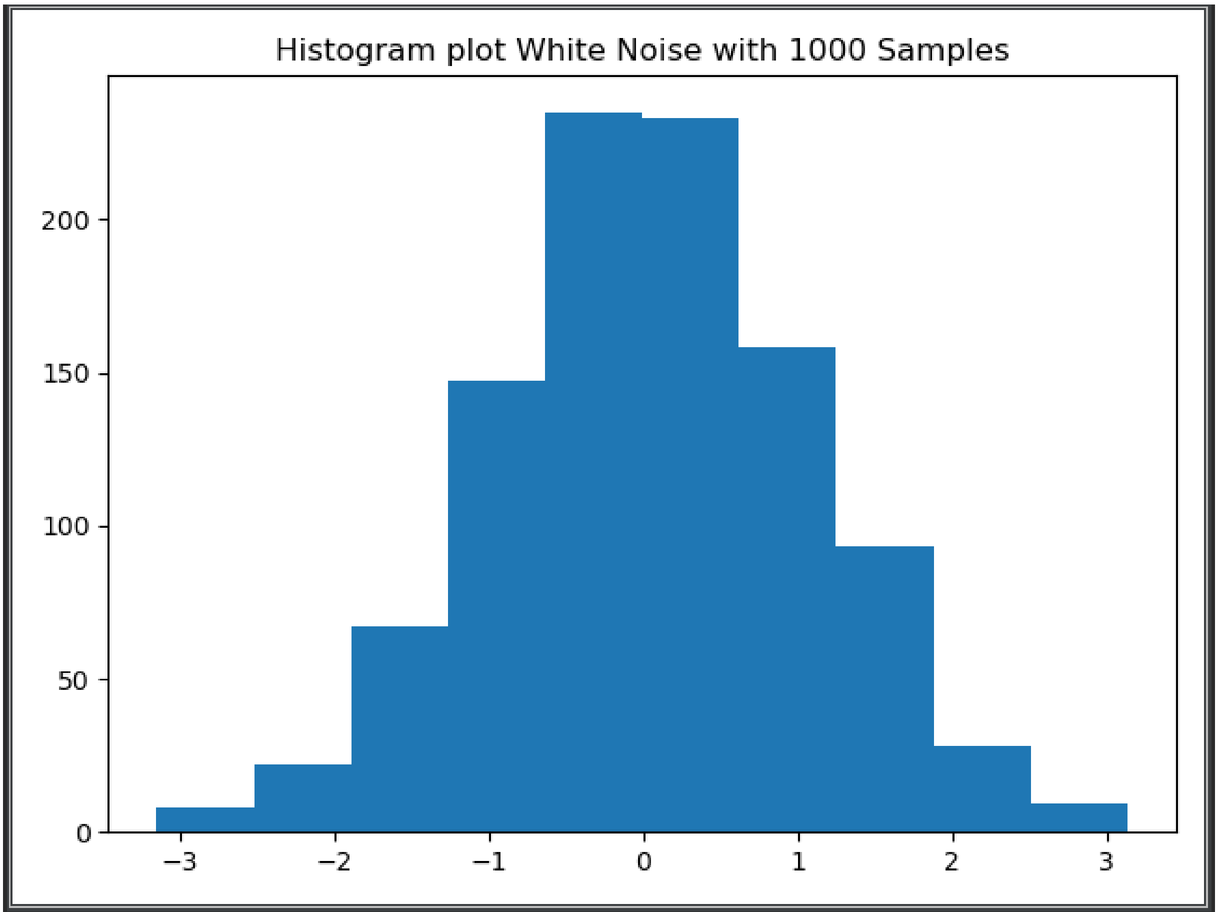
$$= \frac{(y_5-\bar{y})(y_1-\bar{y})}{\sum_{1}^{5} (y_t-\bar{y})^2}$$

$$= \frac{(243-72.6)(3-72.6)}{40075.2}$$

$$= -0.296$$

```
#%%=================================================================================
# 2: Using Python program, create a white noise with zero mean and standard
deviation of 1 and 1000 samples.
# Plot the generated WN versus number of samples.
# Plot the histogram of generated WN.
# Calculate the mean and std of generated WN.
# You can use the following command to generate
# WN~(0,1): (import numpy as np, T # of samples)
# np.random.normal(mean, std, size=T)
#%%=================================================================================
```



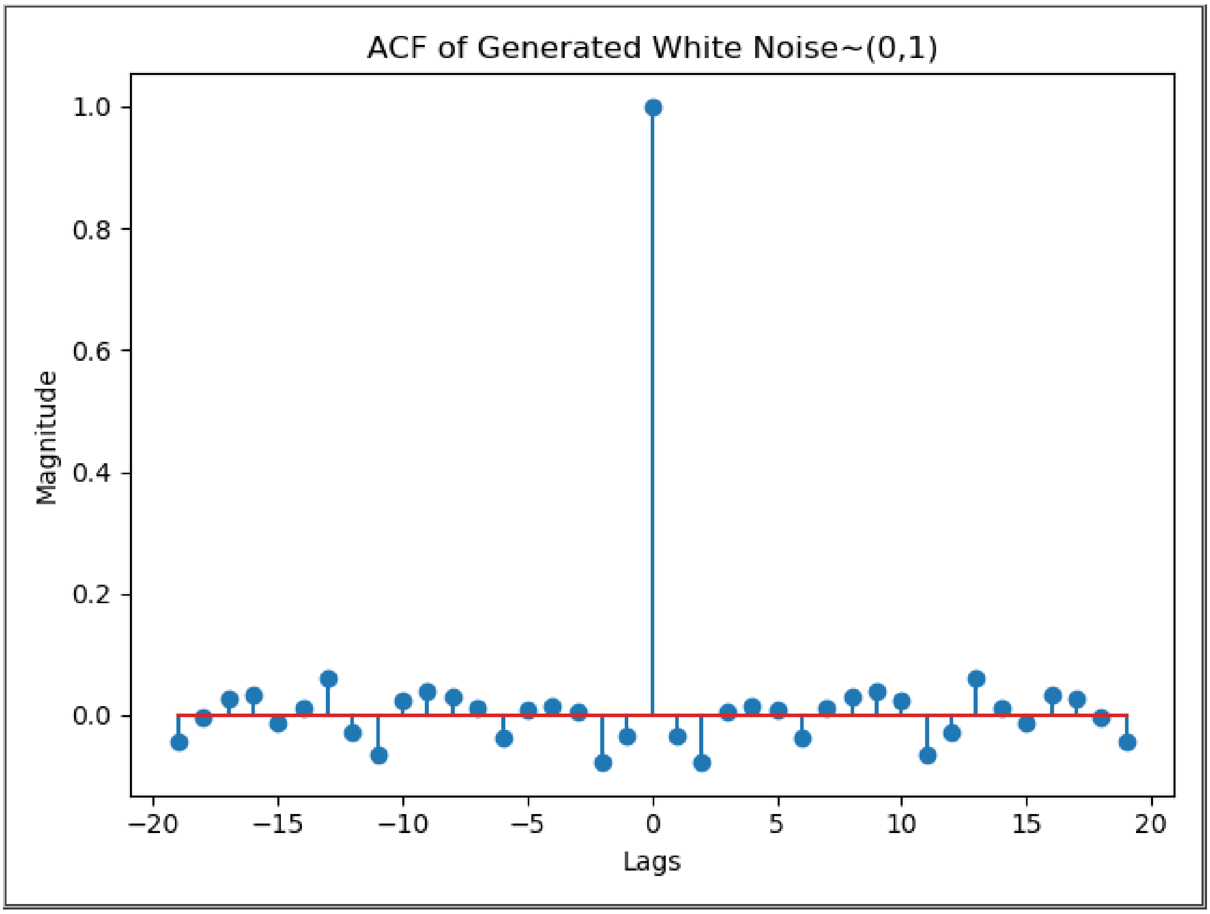



```
The Mean of a white noise: 0.0515396416333361985
The Standard Deviation of White Noise: 1.031411695249564
```

```python
#%%=============================================================================
# 3: Write a python code to estimate Autocorrelation Function.
# Note: You need to use the equation (1) given in lecture 4.
# %%----------------------------------------------------------------------------

def auto_corr(y,k):
    T = len(y)
    y_mean = np.mean(y)
    res_num = 0
    res_den = 0
    for t in range(k,T):
        res_num += (y[t] - y_mean) * (y[t-k] - y_mean)

    for t in range(0,T):
        res_den += (y[t] - y_mean)**2

    res = res_num/res_den
    return res

def auto_corr_cal(y,k):
    res = []
    for t in range(0,k):
        result = auto_corr(y,t)
        res.append(result)
    return res
```

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

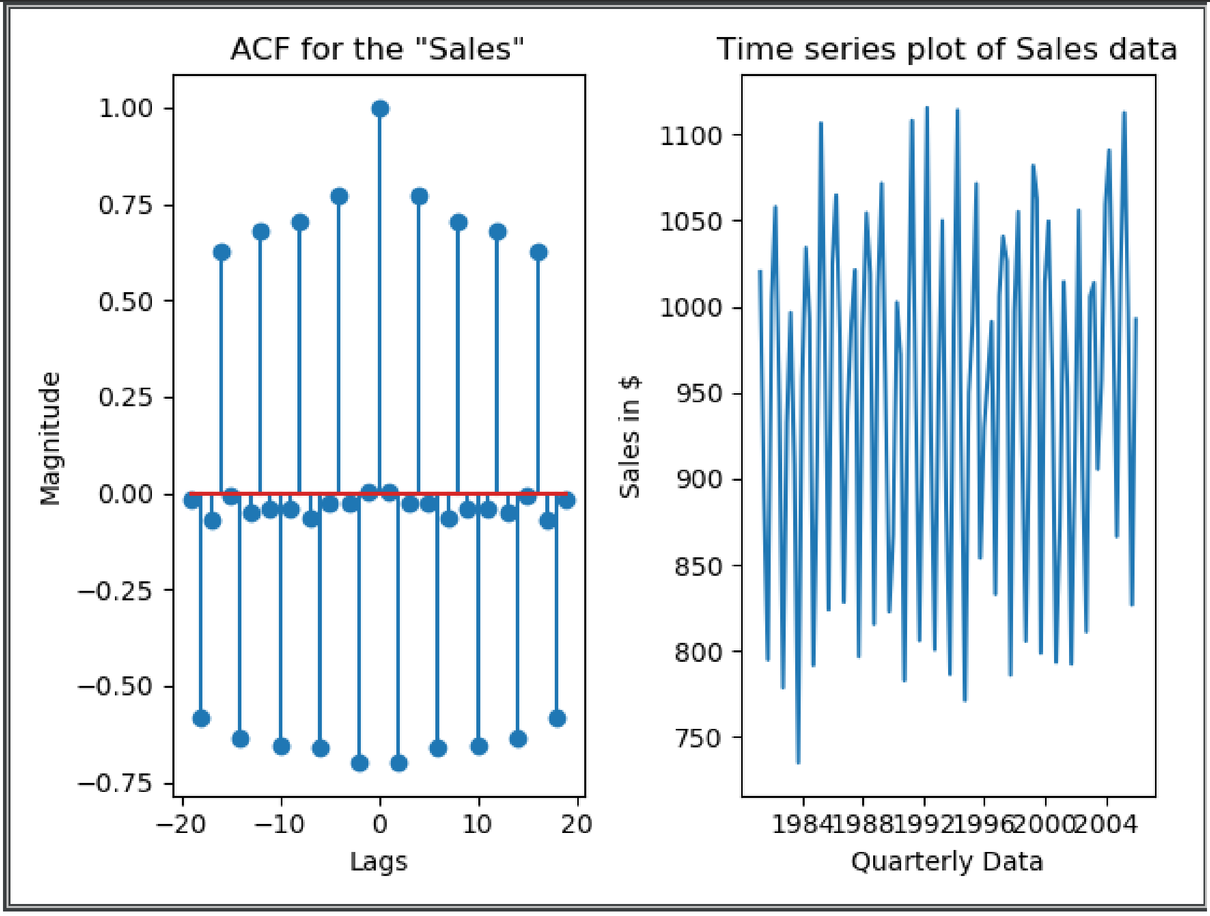
```
#%%=================================================================================
# 3: a. Plot the ACF for the generated data in step 3.
# The ACF needs to be plotted using "stem" command.
# b. Write down your observations about the ACF plot, histogram, and the time plot
of the generated WN
# %%--------------------------------------------------------------------------------
```



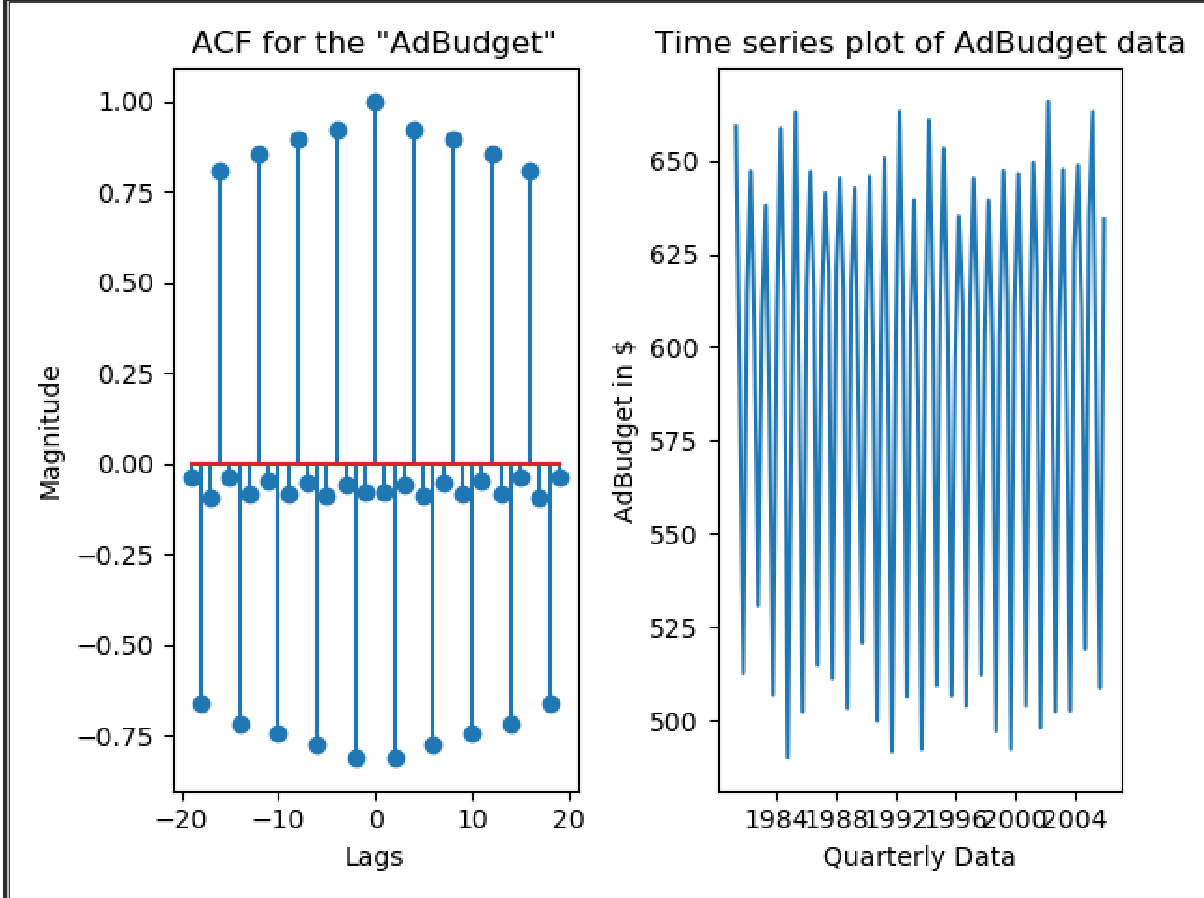

```
Auto correlation of white noise have a strong peak at 0 and absolutely zero for
all other lags which is basically an impulse.

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.
```

```python
#%%===========================================================================
# 4: Load the time series dataset tute1.csv (from LAB#1)
# %%---------------------------------------------------------------------------
k = 20
df = pd.read_csv('tute1.csv')
date_rng = pd.date_range(start='3/1/1981', end='3/1/2006', freq='Q')
```

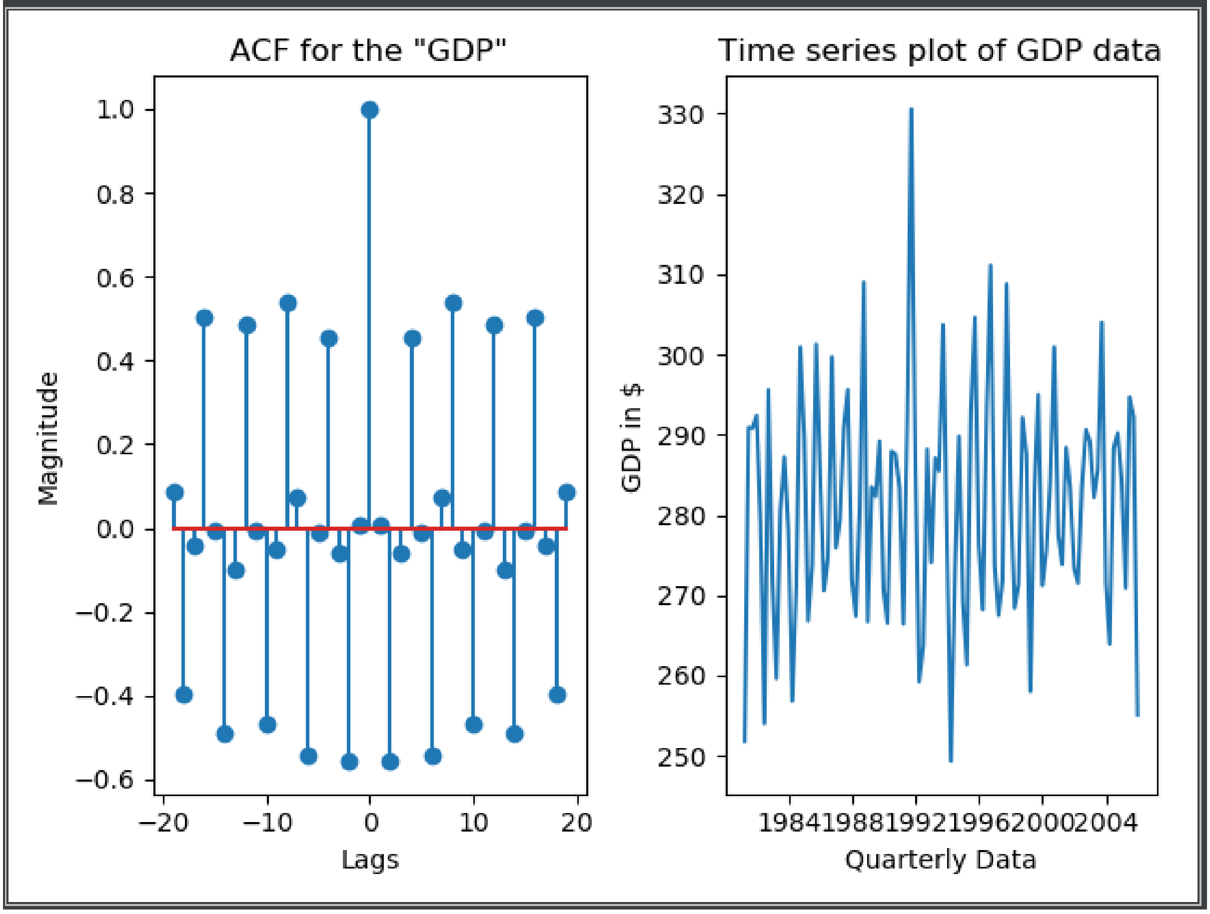
```python
#%%===========================================================================
# 4 a. Using python code written in the previous step,
# plot the ACF for the “Sales” and “Sales” versus time next to each other. You can
use subplot command.
# %%---------------------------------------------------------------------------
```



```
#%%===============================================================================
# 4 b. Using python code written in the previous step,
# plot the ACF for the “Sales” and “Sales” versus time next to each other. You can
use subplot command.
# %%-------------------------------------------------------------------------------
```



```
#%%=================================================================================
# 4 c. Using python code written in the previous step, plot the ACF for the "GDP"
and "GDP"
# versus time next to each other. You can use subplot command.
# %%-------------------------------------------------------------------------------
```



```
#%%===============================================================================
# 4 d. Write down your observations about the correlation between stationary and
nonstationary time series
# (if there is any) and autocorrelation function?
# %%-------------------------------------------------------------------------------
'''
In stationary (time) series, statistical properties such as the mean, variance and
autocorrelation are all
constant over time where as in a non-stationary series statistical properties
change over time.
For a stationary time series, the ACF will drop to zero relatively quickly,
while the ACF of non-stationary data decreases slowly.
'''
#%%===============================================================================
# 4 e. The number lags used for this question is 20.
# %%-------------------------------------------------------------------------------
# Initalized K as 20
```

CONCLUSION

Correlation, auto correlation and their relationship w.r.t stationary and non-stationary data sets are Plotted In stationary (time) series, statistical properties such as the mean, variance and autocorrelation are all constant over time whereas in a non-stationary series statistical property change over time. For a stationary time series, the ACF will drop to zero relatively quickly, while the ACF of non-stationary data decreases slowly.

CHALLENGE

There was no challenge since it was fairly a simple dataset.

APPENDIX

```python
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt


# Number of Samples

T = 1000

# Mean

m = 0

# Standard Deviation

s = 1


Y = np.random.normal(m,s, size=T)

plt.figure()

plt.plot(Y, label = 'White Noise')

plt.xlabel('Number of Samples')

plt.ylabel('Magnitude')

plt.title('White noise with {} Samples'.format(T))

plt.show()


plt.figure()

plt.hist(Y)

plt.title('Histogram plot White Noise with {} Samples'.format(T))

plt.show()


print("The Mean of white noise:",np.mean(Y))

print("The Standard Deviation of White Noise:",np.std(Y))
```

```
def auto_corr(y,k):
    T = len(y)
    y_mean = np.mean(y)
    res_num = 0
    res_den = 0
    for t in range(k,T):
        res_num += (y[t] - y_mean) * (y[t-k] - y_mean)

    for t in range(0,T):
        res_den += (y[t] - y_mean)**2

    res = res_num/res_den
    return res

def auto_corr_cal(y,k):
    res = []
    for t in range(0,k):
        result = auto_corr(y,t)
        res.append(result)
    return res
k = 20
acfcal = auto_corr_cal(Y,k)
acfplotvals = acfcal[::-1] + acfcal[1:]
plt.figure()
plt.stem(range(-(k - 1), k), acfplotvals)
plt.xlabel('Lags')
plt.ylabel('Magnitude')
```

```
plt.title('ACF of Generated White Noise~(0,1)')

plt.show()

k = 20

df = pd.read_csv('tute1.csv')

date_rng = pd.date_range(start='3/1/1981', end='3/1/2006', freq='Q')


Salesacf = auto_corr_cal(df['Sales'],k)

Salesacfplotvals = Salesacf[::-1] + Salesacf[1:]

plt.figure()

fig, (ax1, ax2) = plt.subplots(1, 2)

ax1.stem(range(-(k-1),k), Salesacfplotvals)

ax1.set(xlabel = 'Lags', ylabel = 'Magnitude', title = 'ACF for the "Sales"')

ax2.plot(date_rng, df['Sales'], label = 'Sales')

ax2.set(xlabel = 'Quarterly Data', ylabel = 'Sales in $', title = 'Time series
plot of Sales data')

plt.show()


AdBudgetacf = auto_corr_cal(df['AdBudget'],k)

AdBudgetacfplotvals = AdBudgetacf[::-1] + AdBudgetacf[1:]

plt.figure()

fig, (ax1, ax2) = plt.subplots(1, 2)

ax1.stem(range(-(k-1),k), AdBudgetacfplotvals)

ax1.set(xlabel = 'Lags', ylabel = 'Magnitude', title = 'ACF for the "AdBudget"')

ax2.plot(date_rng, df['AdBudget'], label = 'AdBudget')

ax2.set(xlabel = 'Quarterly Data', ylabel = 'AdBudget in $', title = 'Time series
plot of AdBudget data')

plt.show()
```

```
GDPacf = auto_corr_cal(df['GDP'],k)

GDPacfplotvals = GDPacf[::-1] + GDPacf[1:]

plt.figure()

fig, (ax1, ax2) = plt.subplots(1, 2)

ax1.stem(range(-(k-1),k), GDPacfplotvals)

ax1.set(xlabel = 'Lags', ylabel = 'Magnitude', title = 'ACF for the "GDP"')

ax2.plot(date_rng, df['GDP'], label = 'GDP')

ax2.set(xlabel = 'Quarterly Data', ylabel = 'GDP in $', title = 'Time series plot of GDP data')

plt.show()
```

REFERENCES

https://otexts.com/fpp2/#